検索処理の高速化をめざした

データベーススキーマ設計に関する研究

中西 通雄

平成6年12月15日

検索処理の高速化をめざした

データベーススキーマ設計に関する研究

中西 通雄

平成6年12月15日

(1)

検索処理の高速化をめざした

データベーススキーマ設計に関する研究

中西 通雄

平成6年12月15日

内容梗概

データベースのスキーマは，概念レベル，論理レベル，物理レベルの順に設計される．特に，論理レベルのスキーマの設計手法については，数多くの研究が行われてきたが，データベースに対する検索質問を効率良く処理できるようにするためのスキーマ設計手法については，まだ研究の余地がある．本論文では，検索質問を効率良く処理するためのスキーマ設計手法に関して，まず最初に，関係データベースでの設計手法を理論的に考察し，次に，オブジェクト指向データベースについて，実際の構築経験を基にした設計上の知見を述べる．

関係データベースにおいて，結合を含む検索質問の処理効率を向上させることは重要な問題である．本論文の前半では，定型業務における検索質問の処理効率を向上させるために，データベーススキーマを変形して，結合処理の対象になることが多い基底関係を結合した形で蓄積させることにより，結合処理の数を減らすことを考える．まず，データベーススキーマを変形する２つの操作「複写」と「併合」を定義する.「複写」は ある関係スキーマに属する関数従属を 別の関係スキーマにも重複して持たせる操作であり,「併合」は ２つの関係スキーマを１つにまとめる操作である．データベーススキーマに対してこれらの操作を適用しても，質問を実行したときに新たな不整合を生じないために，質問およびデータベーススキーマが満たすべき必要十分条件を示す．なお，条件の必要性から，その条件を満たす限り併合と複写を適用して得られるデータベーススキーマは，さらに複写や併合を行うと新たに不整合を生じるという意味で最適である．複写や併合を適用することにより得られるデータベーススキーマでは，基底関係を結合した形でデータベースが構成されるので，結果として，結合処理を含んだ検索質問の処理効率を向上させることができる．

後半では，オブジェクト指向データベースのスキーマ設計について述べる．たいていのオブジェクト指向データベース管理システムでは，関係データベー

スと同様に検索処理の性能を向上させるために，文字列の属性値に対するインデックスを作成・利用する機能が提供されている．オブジェクト指向データベースではメソッドが定義できるため，メソッドの返す値もインデックスのように扱えれば，検索処理がさらに高速化できる．すなわち，メソッドがオブジェクト識別子を返す場合にも，あらかじめメソッドを実行しておき，その結果のオブジェクト識別子を保存しておくことにより，検索範囲をせばめることができる．この機能は，特に更新の少ないデータベースにおいて有効である．しかし，これを提供しているデータベース管理システムはない．本論文では，更新の少ないデータの例としてタンパク質構造データをとりあげ，試作したオブジェクト指向データベースについて紹介する．さらに，検索を高速にするための方法，および，この作成経験で得たオブジェクト指向データベースのスキーマの設計についての知見を述べる．

タンパク質の立体構造のデータは，Protein Data Bank (PDB) というデータバンクにファイルとして登録されている．できるだけ簡潔にデータを検索できるようにするために，PDB のデータをデータベース化する試みが行われてきたが，特に最近 2 年間で PDB の容量は約 3 倍の 1 ギガバイトに増加しており，データを効率良く蓄積・検索することもデータベース化する際の重要な課題である．各タンパク質は複数個の鎖で構成されており，各鎖はアミノ酸群で，さらに各アミノ酸は原子群で構成されている．オブジェクト指向データベースを用いると，この構造を自然にデータベーススキーマに反映させることができるが，検索および蓄積の効率を良くするためには工夫が必要である．例えば，アミノ酸系列を検索する質問が多いので，アミノ酸名の系列だけを別に文字列としても蓄積しておき，文字列検索を行うようにする．また，オブジェクト指向データベースでは，クラス階層にうまく適合しないような検索質問に対しては，検索処理の効率が悪くなるという弱点がある．これを補完するために，検索質問やメソッドをあらかじめ実行し，その結果として得られたオブジェク

ト識別子の集合を，オブジェクトとして保存できるようにスキーマを設計した．さらに，一つのタンパク質を構成する原子の数は数千個にもおよび，原子座標を対象とした検索が殆んど無いことなどから，個々の原子をオブジェクトとせずに，まとめて一つのオブジェクトとする方が記憶領域を少なくできる．この結果，試作したデータベースでは，アミノ酸系列などの検索が高速に実行でき，しかも，データベースのサイズをPDBの約半分に押えることができた．また，本論文で述べたスキーマ構成方法は，C++言語に基づいた他のオブジェクト指向データベース管理システムを用いる際にも適用することが可能である．

本論文は，筆者が大阪大学基礎工学部情報工学科で行った研究のうち，データベーススキーマの設計手法に関する研究をまとめたものである．

関連発表論文

1. M.Nakanishi, Y.Katsuyama, M.Ito, and A.Hashimoto: "On Designing Relational Database Schemes for Efficient Query Processing", *Proc. of the third Int. Conf. on Computing and Information*, LNCS# 497, pp.264-275 (May 1991).

2. M.Nakanishi, M.Ito and A.Hashimoto: "An Object-oriented database of Protein Structure Data", *Proc. of the first Int. Conf. on Applications of Databases*, LNCS# 819, pp.336-350 (Jun. 1994).

3. 中西通雄, 葛山善基, 伊藤実, 橋本昭洋："検索処理を高速化するためのデータベーススキーマの設計手法", 信学論, J78-D-I, (1995 年 1 月号に掲載予定) .

その他の発表論文

1. M.Ito and M.Nakanishi: "On Deriving Specialization Constraints over Complex Objects", *Proc. of the third Int. Symposium on Database Systems for Advanced Applications*, pp.335-342 (Apr. 1993).

2. M.Ito, K.Shimizu, M.Nakanishi and A.Hashimoto: "Polynomial-Time Algorithms for Computing Characteristic Strings", *Proc. of the fifth Int. Symp. on Combinatorial Pattern Matching*, LNCS# 807, pp.274-288 (Jun. 1994).

3. M.Ito and M.Nakanishi: "Implication Problem for Specialization Constraints on Databases Supporting Complex Objects", *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E77-A, no. 9 (Sep. 1994).

4. 伊藤実，清水邦保，中西通雄，橋本昭洋："文字列集合における識別文字列を求めるための多項式手続き", 信学論, vol. J77-D-I, no. 8, pp.531-538 (Aug. 1994).

5. M.Nakanishi, M.Hashidume, M.Ito and A.Hashimoto: "A Linear-Time Algorithm for Computing Characteristic Strings", *Proc. of the fifth Int. Symposium on Algorithms and Computation*, LNCS# 834, pp.315-323 (Aug. 1994).

目 次

図目次

表 目 次

第 1 章

緒言

データベースのスキーマは，概念スキーマ，論理スキーマ，物理スキーマの順に設計される．特に，論理スキーマの設計手法については，数多くの研究が行われてきたが，データベースに対する検索質問を効率良く処理できるようにするための，論理スキーマの設計手法については，まだ研究の余地がある．

まず第 2 章では，関係データベースの論理スキーマ設計について述べる．近年，クライアント／サーバ型の情報処理システムの構築が盛んになり，定型業務にも関係データベースがよく用いられるようになってきた [Masu 94]．従来，関係データベースは非定型業務でも利用されることが多く，すべてのデータベース更新に対して不整合が生じないようにするため，データベーススキーマを正規化する必要があった．しかし，データベーススキーマを第 3 正規形（third normal form: 3NF）にすると，関連するデータが複数の関係スキーマに分けられることも多く，検索する際に基底関係を結合（join）する必要がある．そして，この結合処理は，特にレコード数の多いデータベースにおいて，処理時間が長くなるという問題がある．

そこで，結合処理を高速化するために，ソートマージや動的クラスタリングを用いたアルゴリズム，及び，それらを実現した専用のハードウェアを用いる方法が提案されている [Kitsu 91]．また，基底関係にインデックスを設定して結合処理を高

速化する方法も，従来からよく用いられている．

本論文では，定型業務と 3NF のデータベーススキーマが与えられた時，定型業務における検索質問の処理効率を向上させるために，データベーススキーマを変形して，結合処理の対象になることが多い基底関係を結合した形で蓄積させることにより，結合処理の数を減らすことを考える．まず，データベーススキーマを変形する２つの操作「複写」と「併合」を定義する.「複写」は ある関係スキーマに属する関数従属を 別の関係スキーマにも重複して持たせる操作であり,「併合」は ２つの関係スキーマを１つにまとめる操作である．データベーススキーマに対してこれらの操作を適用しても，質問を実行したときに新たな不整合を生じないために，質問およびデータベーススキーマが満たすべき必要十分条件を示す．なお，条件の必要性から，その条件を満たす限り併合と複写を適用して得られるデータベーススキーマは，さらに複写や併合を行うと新たに不整合を生じるという意味で最適である．複写や併合を適用することにより得られるデータベーススキーマでは，基底関係を結合した形でデータベースが構成されるので，結果として，結合処理を含んだ検索質問の処理効率を向上させることができる．本論文では，さらに，この必要十分条件が成立しない場合にも，データベースの機能や定型業務の処理内容によっては，複写や併合が適用できる場合があることを示す．また，複写や併合が利用されている例として，ある会社の実システムを紹介し，その定型業務とデータベーススキーマを分析する．

複写または併合を行うと，データベーススキーマが 3NF から 2NF もしくは，1NF になる．検索性能を向上させるために，1NF をさらに非正規化する手法も実際のデータベースシステムで用いられるが，本論文では扱わない．一方，検索性能を向上させるという観点ではないが，現実世界のデータをなるべく自然にモデル化するためのアプローチとして，スキーマを必ずしも 1NF にもする必要がないという主張が牧之内によってなされている [Maki 77]．そこでは，関数従属とは異なった従属関係を新たに導入し，1NF ではないがある種の正規形にするという議論が展開さ

れている．また，このアプローチは，関係データベースのスキーマについて論じたものであるが，オブジェクト指向に基づくモデル化の方法として引用されることも多い．

第3章では，オブジェクト指向データベースのスキーマ設計について述べる．最近，科学技術データに着目したデータベースの研究が盛んになりつつある [TCDE 93, Tana 93]．扱うデータとしては，分子生物学（遺伝子，タンパク質），化学（化合物，化学反応），地球科学（気象，地質，地震，環境），宇宙科学（天文，衛星），医学（遺伝病，臨床例）などがある．これらのデータには，

1. データ量が膨大
2. 構造が複雑
3. データソースの媒体（CD-ROM, テープ, 光磁気ディスク等）が多様
4. データは追加だけで更新が殆んど無い

などの特徴があり，これを適切に処理することがデータベースの要件になっている．科学技術データベースをサポートするためのデータモデルとしては，従来の関係データベースを拡張した拡張関係データベース，オブジェクト指向データベース，または，それらの組合せが有望である [Shos 92]．

科学技術データのなかでも，特に分子生物学の分野では，ヒトを始めとして，大腸菌，マウス，酵母，枯草菌，イネなどの遺伝子を解析するプロジェクトが世界中で協力しあって進められており，DNA の塩基配列を読み取る機器の進歩やコンピュータの利用とあいまって，遺伝子やタンパク質のデータ量はすさまじい勢いで伸びてきている．そこで，これらをデータバンク（表 1.1）として作成し，ネットワークを通して利用できるようにする仕事が進められて来た [Taka 92, Erick 92]．遺伝子やタンパク質のデータは，従来からテキスト形式のファイルで蓄積されており，現在でもなお，このテキスト形式で使われているのがほとんどである．関係データベース管理システムやオブジェクト指向データベース管理システムを用いて，遺伝子データをデータベース化する研究はいくつか行われている．しかし，ほとんどが研究レ

表 1.1 分子生物学関連の主なデータバンク

名称	内容	収集機関
GENBANK	遺伝子の塩基配列	Los Alamos National Laboratory
EMBL	〃	EMBL
DDBJ	〃	国立遺伝研
GDB	ヒト遺伝子地図	Johns Hopkins University
PIR	タンパク質のアミノ酸配列	National Biomedical Research Foundation
SWISS-PROT	〃	Centre Medical Universitaire
PDB	タンパク質の立体構造	Brookhaven National Laboratory
MEDLINE	医学文献	National Library of Medicine

ベルに留まっており，筆者の知る限り，本格的に運用しているのは GDB だけである．GDB では，SYBASE を用いて 300 個を超えるテーブル（関係）を作成し，ヒトの遺伝子データを保存している．

筆者の研究グループでは，分子生物学データの一例としてタンパク質の立体構造のデータを対象として取り上げ，データベースを試作した．タンパク質の立体構造のデータは，タンパク質ごとに FORTRAN カード形式でファイルとして作成され，Protein Data Bank (PDB) というデータバンクに登録されている．図 1.1に示すように，特に最近 2 年間で，PDB の容量およびエントリ数は約 3 倍になっており，現在，容量は 1GB で，約 3000 種のタンパク質の構造データが含まれている．できるだけ簡潔にデータを検索できるようにするために，PDB のデータをデータベース化する試みが行われてきたが，データを効率良く蓄積・検索するという点も重要な課題である．関係データベースを用いて実現した例では，

1. 複雑な関連を表現しにくい．
2. 系列を表現しにくい．
3. SQL で複雑な計算を記述しにくい．また，画像表示などの応用プログラムを呼び出せない．

などの欠点があった．そこで，最近，オブジェクト指向データベースを用いて構築

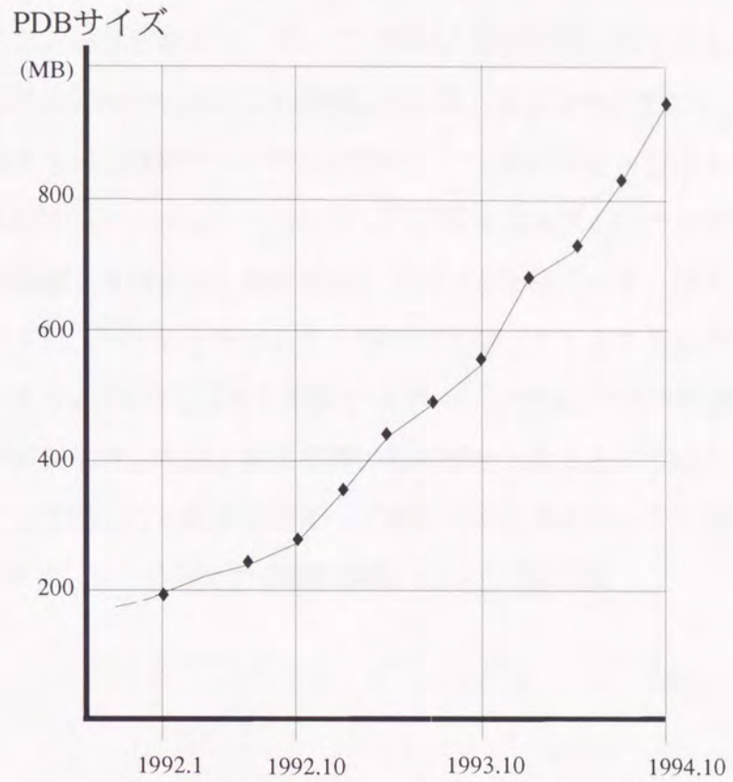


図 1.1 PDB のデータ量

する研究も進められてきている．本論文では，商用のオブジェクト指向データベース管理システムを用いて作成したタンパク質の構造データのデータベースについて紹介し，この作成経験で得たデータベーススキーマの設計についての知見を述べる．

各タンパク質は，複数個の鎖で構成されており，鎖はアミノ酸群で，さらにアミノ酸は原子群で構成されている．また，鎖を構成するアミノ酸の部分系列は，らせんなどの 2 次構造と呼ばれる立体構造をとる． オブジェクト指向データベースでは，この構造を自然にデータベーススキーマとして反映させることができる．例えば，タンパク質オブジェクトは，鎖を表す複数のオブジェクトで構成し，鎖は 2 次構造で，さらに 2 次構造は，複数個のアミノ酸のオブジェクトで構成すればよい．ところが，タンパク質データベースに対する検索には，あるアミノ酸の系列を部分系

列として含むタンパク質を捜す質問が多く．この質問を処理するためには，数百個のアミノ酸をたどる必要がある．従って，質問の処理効率を良くするためには，アミノ酸系列だけを別に文字列として蓄積しておき，文字列検索を行えばよい．また，個々の原子は3次元の座標データを持つため，一つの原子を一つのオブジェクトで表すのが自然である．しかし，一つのタンパク質を構成する原子の数は数千個にもおよび，原子座標を対象とした検索が殆んど無いことなどから，個々の原子をオブジェクトとせずに，ひとまとめにして一つのオブジェクトとする方が記憶領域が少なくて済む．さらに，オブジェクト指向データベースでは，クラス階層にマッチしないような検索質問に対しては，検索処理の効率が悪くなるという弱点があるが，本論文では，これを補完するためのスキーマ設計上の工夫についても述べる．

最後に4章で，まとめと今後の研究課題について述べる．

第 2 章

関係データベースのスキーマ設計

2.1 まえがき

関係データベースでは，すべてのデータベース更新に対して不整合が生じないようにするため，データベーススキーマを正規化する必要がある．しかし，データベーススキーマを第 3 正規形（3NF）にすると，関連するデータが複数の関係スキーマに分けられることも多く，検索する際に基底関係を結合（join）する必要がある．そして，この結合処理は，特にレコード数の多いデータベースにおいて，処理時間が長くなるという問題がある．そこで，結合処理の対象になることの多い基底関係をあらかじめ結合してデータベースに格納し，結合処理そのものを減らすようにデータベーススキーマを設計する手法が非正規化の一手法として提案され，実際のデータベースを構築する場合に用いられてきた [Kirk 92, Inoue 93, NTT 93] ．これは，定型業務においては質問で用いられるデータベースへのアクセス経路が固定しているので，アクセス経路を分析することにより，3NF ではないが不整合を生じないようなデータベーススキーマを設計できる場合があるからである．

非正規化することにより検索処理性能が向上するような基底関係や属性を見つけるために，基底関係や属性に対するアクセス頻度を計算する方法などが文献

[Inmon 87, Inmon 89] で示されている．しかし，見つけた属性に対して非正規化を適用した後に不整合を生じることがないかどうかについては，今まで形式的な議論がなされておらず，その保証はシステムを熟知している設計者の勘に頼るところが大きかった．

本論文では，3NF のデータベーススキーマと定型業務が与えられた時，基底関係を結合した形で格納する方式について述べる．まず，データベーススキーマを変形する２つの操作として「複写」と「併合」を定義する.「複写」はある関係スキーマに属する１つの関数従属を別の関係スキーマにも重複して持たせる操作であり,「併合」は２つの関係スキーマを１つにまとめる操作である．そして，複写あるいは併合を行っても，定型業務の変更（挿入，削除，または更新）質問が新しいデータベーススキーマのもとで新たな不整合を生ぜず，かつ，検索結果が等価になるように実現できるための必要十分条件を示す．なお，一貫性制約としては関数従属を考える．

本論文で提案する（データベース）スキーマの変形方法を用いることにより得られる新しいデータベーススキーマ上では，結合処理の数を減らせるため，定型業務における検索質問の処理効率を向上させることができる．

2.2 諸定義

関係データベースの基本概念については，文献 [Maier 83] 等をもとにしており，ここでは本論文で必要なものだけを定義する．

関係スキーマを $\langle R, F\rangle$ で表す．但し，R は属性集合，F は R 上の関数従属の集合である．$\langle R, F\rangle$ 上の関係 r とは，F に属するすべての関数従属を満たす R上で定義される関係である．Fに属するすべての関数従属を満たす関係が常に満たす（すなわち，Fから導出される）関数従属の集合を F^+ と表す．

関係スキーマの系列 $\mathbf{R}$:$\langle R_1, F_1\rangle, \cdots, \langle R_n, F_n\rangle$ をデータベーススキーマと言う．$\mathbf{R}$ 上のデータベースとは，関係の系列 $\mathbf{D}$: $r_1, r_2, ..., r_n$ で，各 r_iが $\langle R_i, F_i\rangle$ 上の関係で

あるものを言う．$\mathbf{D}$ に属する関係 r_iを基底関係という．以下，基底関係はどの属性にも未定義値を許さないと仮定する．

関係 r_iと関係 r_jの結合 (natural join) を $r_i \bowtie r_j$ と表し，関係 r_iの属性集合 X上への射影を $\pi_X(r_i)$ と表す．データベース $\mathbf{D}$: $r_1, r_2, ..., r_n$を変更する操作は，次の 3 種類である．

1. 挿入： データベース $\mathbf{D}$: $r_1, \cdots, r_n$に対する $R_1 \cup \cdots \cup R_n$の部分集合上の組τ の挿入とは，R_iの属性値が 1 つ以上τ に含まれているような各基底関係 r_iに対して，r_i を $r_i \cup \{\tau[R_i]\}$ で置換することである．未定義の属性値を持つ組の挿入は，後で定義する不整合となりうる．また，$r_i \cup \{\tau[R_i]\}$ において，F_i が満たされなければ，その挿入は拒否される．ここで，$\tau[R_i]$ は τ の R_i成分の属性値を表す．また，$\tau[R_i]$ を τ の R_i属性値と呼ぶ．

2. 削除： データベース $\mathbf{D}$: $r_1, \cdots, r_n$に対する $R_1 \cup \cdots \cup R_n$の部分集合上の組τ の削除とは，R_i の属性値が 1 つ以上 τ に含まれているような各基底関係 r_iに対して，r_i を $r_i - T_i$ で置換することである．ここでT_iは次式で定義される組の集合である．但し，S_iは，R_i の部分集合で，かつ，$\tau[R_i]$ の中で値が定義されている属性の集合とする．

$$T_i = \{\tau_i \in r_i \mid \tau_i[S_i] = \tau[S_i]\}$$

ここで，$r_i - T_i$ は，F_i を自動的に満たす．

3. 更新： データベース $\mathbf{D}$: $r_1, \cdots, r_n$に属する $R_{i_1} \cup \cdots \cup R_{i_p}$上のある組$\tau$ に対して，$\tau[S]$ 値をτ' 値に更新するとは，以下の処理を行なうことである． $(1 \leq i_1, \cdots, i_p \leq n)$

   各基底関係 r_{i_q} $(1 \leq q \leq p)$ を
   $(r_{i_q} - \{\tau[R_{i_q}]\}) \cup \{\tau_{i_q}\}$ で置換する．

但し，S は更新したい属性の集合で $R_{i_1} \cup \cdots \cup R_{i_p}$の部分集合とする．$\tau'$ は S 上の組であり，S_{i_q}は $S_{i_q} = R_{i_q} \cap S$を満たす属性集合とする．また，τ_{i_q}は，$\tau_{i_q}[S_{i_q}] = \tau'[S_{i_q}]$ かつ $\tau_{i_q}[R_{i_q} - S_{i_q}] = \tau[R_{i_q} - S_{i_q}]$ を満たす R_{i_q}上の新しい組とする．

二つの基底関係にまたがる検索は，二つの基底関係の結合をとるものとする．即ち，$S \subseteq R_j \cup R_k$とするとき，Sに対する検索とは，$\pi_{S \cap R_j}(r_j) \bowtie \pi_{S \cap R_k}(r_k)$ を検索するものとする．

質問は，検索質問または変更（挿入，削除，または更新）質問とする．質問 Q が $\langle R_i, F_i \rangle$ 上の基底関係 r_iを操作の対象に含むとき，質問 Q は $\langle R_i, F_i \rangle$ を使う，または r_iを使うと言う．

不整合とは，データベースに対して変更操作を行うときに起り得る望ましくない性質のことである．次に，本論文で対象とする 3 つの不整合を定義する．(各不整合の例は，定義の後に与えられる．) 変更操作は実質的に個々の基底関係に対して行われるので，不整合をまず 1 つの基底関係に対して定義する．以下，$\mathbf{D}$ を $\mathbf{R}$ のデータベース，$\langle R, F \rangle$ を $\mathbf{R}$ に属する一つの関係スキーマとする．

1. 挿入不整合：挿入は基底関係に対して一つの組単位で行うので，挿入する組が未定義の属性値を含むとき，その属性値を決められなければ，挿入により不整合を生じることになる．次の条件が共に成立するとき，$\mathbf{D}$ において挿入不整合が生じるという．

   (a) $\langle R, F \rangle$ 上の基底関係 rに対して，$X \subset R$ 属性値を挿入する要求がある．但し，$X \subset R$ は Xが Rの真部分集合であることを示す．

   (b) 挿入したい組の $R - X$属性値を，$\mathbf{D}$ から追跡法（chase）を用いて計算できない．[1]

[1] 追跡法の詳細な定義は文献 (Maier 83) の 8.6 節を参照．直観的には，各関数従属 $Z \rightarrow W \in F$ を組の Z属性値からその組の W属性値への関数とみなし，その関数に基づいて未定義の属性値を計算する．

2. 削除不整合：削除は基底関係から組の集合を取除くが，取り除く組がたとえ一つであっても，その組の一部分の属性値を削除したいときには，削除の対象と考えていた以上のデータを削除してしまう可能性がある．次の条件が共に成立するとき，**D** において削除不整合が生じるという．

   (a) $\langle R, F\rangle$ 上の基底関係 rに対して，ある組τ の $X \subset R$属性値を削除する要求がある．

   (b) 削除操作の定義に従って r から組τを削除することになり，その結果，$\pi_{R-X}(r) \neq \pi_{R-X}(r - \{\tau\})$ となる．即ち，τ の $(R - X)$ 属性値が **D** から失われる．

3. 更新不整合：ある一つの組の属性値を変更すると，関数従属の制約を満たすようにするために，他の組の属性値も変更しなければならない場合がある．次の条件が共に成立するとき，**D** において更新不整合が生じるという．

   (a) $\langle R, F\rangle$ 上の基底関係 rに対して，R上の組τ の一部またはすべての属性値を更新する要求がある．

   (b) τ の更新に伴い，Fを満たすために更新の必要な組がτ 以外に一つ以上存在する．

[例 1] 図 2.1の関係「講義」を考える．関数従属は {科目→教官, 教官→教官室} とする．

1. 挿入不整合：「鈴木」と「$A203$」という従属関係の情報（以下，単に情報と呼ぶ）を新たに挿入すると，担当の科目が未定義となるので不整合を生じる．一方,「データベース」と「山田」という情報ならば，既に「$A101$」という教官室が決っているので不整合を起こさずに挿入できる．

2. 削除不整合：「アルゴリズム」という科目が無くなったとき，その科目を削除したいとする．削除操作の定義に従って組τ_3 を削除すれば,「山田」と「$A101$」

	科目	教官	教官室
τ_1 :	オートマトン	佐藤	$A105$
τ_2 :	コンパイラ	佐藤	$A105$
τ_3 :	アルゴリズム	山田	$A101$

図 2.1 関係「講義」

という情報が失われてしまい不整合を生じる．一方,「コンパイラ」という科目が無くなったとき組τ_2を削除しても，組τ_1に「佐藤」と「$A105$」という情報は残るので不整合は生じない．

3. 更新不整合 :「オートマトン」を講義している「佐藤」教官の教官室を「$A105$」から「$A104$」に変更したいとする.「佐藤」と「$A105$」を含む組はτ_1とτ_2であるため,「教官→教官室」という関数従属を満たすためにはこの 2 つの組の更新が必要となり，不整合を生じる． □

データベース **D** に対する変更質問が，**D** のどれか 1 つの基底関係に対して不整合を生じるとき，その質問は不整合を生じると言う．また，定型業務を構成する質問の中で 1 つでも不整合を生じる質問があるとき，その定型業務は不整合を生じると言う．

3NF のデータベーススキーマと定型業務の集合が与えられ，どの定型業務も不整合を生じないとする．ここで，定型業務とは，そこで用いられる質問が固定していて，各質問のアクセス経路も決っているものとする．

2.3 関数従属の複写

2.3.1 複写の定義

定型業務の質問では，関連する２つのスキーマ上のデータを検索したいとき，基底関係の結合処理を行う．そこで，ある基底関係の射影と別の基底関係を結合するような検索質問が多い場合には，あらかじめ基底関係を結合してデータベースに格納することにより，検索質問の処理時間を短縮できる．基底関係を結合させた形で蓄積するために，与えられたデータベーススキーマを変形することを考える．変形の操作として，関係スキーマ中の１つの関数従属を別の関係スキーマに重複させた関係スキーマを作る操作を次のように定義する．

[定義 1] データベーススキーマ $\mathbf{R}$ に属する２つの関係スキーマ $\langle R_j, F_j \rangle$, $\langle R_k, F_k \rangle$ に対して，$X \subseteq R_j \cap R_k$かつ $Y \not\subseteq R_j$ を満たす関数従属 $X \rightarrow Y$ が F_kに存在するとする．このとき，関数従属 $X \rightarrow Y$を $\langle R_j, F_j \rangle$ に複写するとは，$\mathbf{R}$ において $\langle R_j, F_j \rangle$ を $\langle R_j \cup Y, F_j \cup \{X \rightarrow Y\} \rangle$ で置換することを言う．但し，$\langle R_k, F_k \rangle$ は変更せずにそのまま残す． □

[例 2] 図 2.2に示すように，3NF の関係スキーマが 2 つ与えられ，$X \rightarrow Y$に相当する FD があるものとする．この例では，関係スキーマの属性名だけを [ ] 内に示している．「FD:商品番号→ 商品名」を複写するとは，「商品名」という属性と「FD:商品番号 → 商品名」を受注データのスキーマに追加した新受注データというスキーマ作り，受注データのスキーマを置換することである．「商品データ」のスキーマは変更せずにそのまま残す． □

以下，簡単のために，関係スキーマ $\langle R_j, F_j \rangle$, $\langle R_k, F_k \rangle$ をそれぞれ $\langle R_1, F_1 \rangle$, $\langle R_2, F_2 \rangle$ とする．

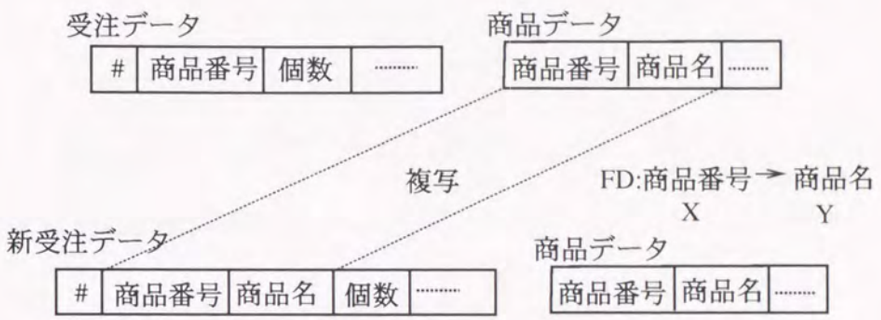


図 2.2 スキーマ複写の例

2.3.2 検索質問の変換定義

データベーススキーマ $\mathbf{R}$ に属する2つの関係スキーマ $\langle R_1, F_1\rangle, \langle R_2, F_2\rangle$ に対して，$X \to Y \in F_2$ を $\langle R_1, F_1\rangle$ に複写して作成したデータベーススキーマを $\mathbf{Rt}$ とする．$\mathbf{R}$ 上のデータベースを $\mathbf{D} : r_1, r_2, \cdots, r_n$ とし，$\mathbf{Rt}$ 上のデータベースを $\mathbf{Dt} : r_{12}, r_2, \cdots, r_n$ とする．ここで，r_1, r_2, r_{12} は，それぞれ $\langle R_1, F_1\rangle$, $\langle R_2, F_2\rangle$, $\langle R_1 \cup Y, F_1 \cup \{X \to Y\}\rangle$ の上の基底関係とする．

関数従属を複写する目的は，$(R_1 \cup Y)$ に対する検索が多いので，r_1 のかわりに $r_1 \bowtie \pi_{X \cup Y}(r_2)$ を r_{12} として記憶し，検索効率を上げることである．従って，複写に伴い，検索質問を定義2に示すように変換する．

[定義 2] F_2 に属する関数従属 $X \to Y$ の複写に伴って，$\langle R_1, F_1\rangle$ または $\langle R_2, F_2\rangle$ を使う検索質問を次のように変換する．

r_1 と $\pi_{X \cup Y}(r_2)$ を結合して得られる関係 $r_1 \bowtie \pi_{X \cup Y}(r_2)$ に対する検索は，r_{12} に対する検索に変換する．この結合以外の検索は，r_1 を $\pi_{R_1}(r_{12})$ で置き換える． □

2.3.3 検索結果が等価であるための必要十分条件

定義2より明らかなように，複写後のデータベース $\mathbf{Dt}$ での検索結果が，複写前のデータベース $\mathbf{D}$ での検索結果と等価であるための，必要十分条件は，式 (2.1) (2.2)

が共に成立することである.

$$r_{12} = r_1 \bowtie \pi_{X \cup Y}(r_2) \tag{2.1}$$

$$\pi_{R_1}(r_{12}) = r_1 \tag{2.2}$$

式 (2.1)(2.2) から次式が導ける.

$$r_1 = \pi_{R_1}(r_1 \bowtie \pi_{X \cup Y}(r_2)) \tag{2.3}$$

[補題 1] 複写後のデータベースにおいても，検索結果が等価となるための必要条件は,

$$\pi_X(r_1) \subseteq \pi_X(r_2) \tag{2.4}$$

が成立することである.

（証明）対偶命題を証明する．式 (2.4) が成立しないと仮定したとき，$\tau[X] \notin \pi_X(r_2)$ であるような組τが r_1 に存在するので，$\pi_{R_1}(r_1 \bowtie \pi_{X \cup Y}(r_2))$ には τ が含まれなくなり，式 (2.3) が成立しない. □

2.3.4 等価が保証されるための変更質問の条件

もとのデータベース **D** に対する検索質問の結果と等価な結果を，変換された検索質問が複写後のデータベース **Dt** から得るためには，**D** に対する変更質問がある条件を満たさなければならない．すなわち，**D** において，ある変更質問の結果，基底関係 r_1, r_2 が r_1', r_2' になったとすると，$\pi_X(r_1') \subseteq \pi_X(r_2')$ が必ず成り立たねばならない．このための必要条件を次の補題で示す.

[補題 2] 補題 1の式 (2.4) が常に成立するための必要条件は，**D** に対するどの変更質問も以下の条件をすべて満たすことである.

条件 1: r_1に対する挿入は，$R_1 \cup R_2$上の属性値を含むものに限られる．すなわち，r_1に対する挿入はr_1, r_2両方に対して同時に行われ，r_1のみに対する挿入はない.

条件 2: r_2に対して削除がない.

条件 3: r_1の X属性値は更新しない.

(証明) 条件 1が成立しない場合：R_2属性値を含まない組の挿入がある. 例として, r_1, r_2がともに空の状態の時に, R_1上の組τを挿入して得られるデータベースを考えると, 式 (2.4) が成立しない.

条件 2が成立しない場合：例として, r_1, r_2がともに空の状態で, $R_1 \cup R_2$上の組 τ を挿入した後, r_2に挿入された組$\tau[R_2]$ を削除する操作で得られるデータベースを考える. この時, 式 (2.4) が成立しなくなる.

条件 3が成立しない場合: r_1に属するある組の X属性値に更新がある. 例として, r_1, r_2がともに空の状態で, $R_1 \cup R_2$上の組τを r_1, r_2に挿入（すなわち r_1に$\tau[R_1]$ を挿入し, r_2に$\tau[R_2]$ を挿入）した後, r_1に挿入された組τ の X属性値を前の値と異なる値に更新する操作で得られるデータベースを考える. この時, 式 (2.4) が成立しなくなる. □

2.3.5 変更質問の変換定義

関数従属の複写に伴い, もとのデータベーススキーマにおける変更質問を新しいデータベーススキーマにおける質問に変換する必要がある. $\langle R_1, F_1\rangle$ または $\langle R_2, F_2\rangle$ を使う質問 Q を考える. 即ち, Q は, $\mathbf{R}$ 上のデータベース $\mathbf{D}: r_1, r_2, \cdots, r_n$ を $\mathbf{D}'$: $r'_1, r'_2, \cdots, r'_n$に変更（挿入, 削除, または更新）する質問とする. 図 2.3に示すように, 関数従属の複写に伴い, Q を新しいデータベーススキーマにおける質問 $T(Q)$ に変換する必要がある. スキーマを変換しても等価な検索結果が得られることが保証されるような変更質問のクラスに対してだけ考えればよいため, 定義 3では, 補題 2の条件 1,2,3がすべて成立する場合についてのみ定義する.

[定義 3] データベーススキーマ $\mathbf{R}$ に属する 2 つの関係スキーマを $\langle R_1, F_1\rangle, \langle R_2, F_2\rangle$ とし, その上の基底関係をそれぞれ r_1, r_2とする. また, r_{12}を $\langle R_1 \cup Y, F_1 \cup \{X \rightarrow Y\}\rangle$

		Q				$T(Q)$	
R:	$\mathbf{D}$	$\Longrightarrow$	$\mathbf{D}'$	Rt:	$\mathbf{Dt}$	$\Longrightarrow$	$\mathbf{Dt}'$
	r_1		r'_1		r_{12}		r'_{12}
	r_2		r'_2		r_2		r'_2
	.		.		.		.
	.		.		.		.
	r_n		r'_n		r_n		r'_n

図 2.3 変更操作による基底関係の対応

上の基底関係とする．F_2に属する関数従属 $X \to Y$の複写に伴って，$\langle R_1, F_1 \rangle$ または $\langle R_2, F_2 \rangle$ を使う変更質問を次のように変換する．

1. 挿入：データベース $\mathbf{D}: r_1, \cdots, r_n$ に対する $R_{i_1} \cup \cdots \cup R_{i_p}$上の組$\tau$の挿入は次の 2 通りだけを考えればよい[2]．$(1 \leq i_1, \cdots, i_p \leq n)$

   (a) $1 \notin \{i_1, \cdots, i_p\}$ の場合：変換前の操作をそのまま行う．

   (b) $1 \in \{i_1, \cdots, i_p\}$ かつ $2 \in \{i_1, \cdots, i_p\}$ の場合：$\tau[R_1]$ を r_1へ挿入する操作を，$\tau[R_1 \cup Y]$ を r_{12}に挿入する操作に変換する．r_2への挿入は，変換前と同じ（$\tau[R_2]$ を r_2に挿入する操作）である．

2. 削除：データベース $\mathbf{D}: r_1, \cdots, r_n$ に対する $R_{i_1} \cup \cdots \cup R_{i_p}$上の組$\tau$の削除は，$1 \in \{i_1, \cdots, i_p\}$ かつ $2 \notin \{i_1, \cdots, i_p\}$ の場合のみ考えれば良い．即ち，r_1に対する削除は，r_{12}において$\tau_{12}[R_1] = \tau[R_1]$ を満たすすべての組τ_{12} を削除する操作に変換する．ここで，r_1, r_2, r_{12}に対する削除の結果を r'_1, r'_2, r'_{12}とするとき，$r'_1 = \pi_{R_1}(r'_{12})$ が成り立つように，上記に該当するすべての組を削除する．

[2]与えられた定型業務は不整合を生じないという前提から，R_1の真部分集合上の組の挿入・削除を考える必要はなく，挿入・削除の対象となる組τの R_1属性値はすべて与えられるものとしてよい．

3. 更新：データベース $\mathbf{D}: r_1, \cdots, r_n$ に対する $R_{i_1} \cup \cdots \cup R_{i_p}$ 上の組 τ の更新は，次の 3 通りに分類できる．

   (a) $1 \in \{i_1, \cdots, i_p\}$ かつ $2 \notin \{i_1, \cdots, i_p\}$ の場合：r_1 に対する更新は，r_{12} において $\tau_{12}[R_1] = \tau[R_1]$ を満たすすべての組 τ_{12} を更新する操作に変換する．ここで，r_1, r_{12} に対する更新の結果を r'_1, r'_{12} とするとき，$r'_1 = \pi_{R_1}(r'_{12})$ が成り立つように，該当するすべての組を更新する．

   (b) $1 \notin \{i_1, \cdots, i_p\}$ かつ $2 \in \{i_1, \cdots, i_p\}$ の場合：r_2 に対する更新は，r_2 において $\tau[R_2]$ を更新する．更に r_{12} において $\tau_{12}[X \cup Y] = \tau[X \cup Y]$ を満たすすべての組 τ_{12} を更新する操作に変換する．ここで，$r'_{12} = r'_1 \bowtie \pi_{X \cup Y}(r'_2)$ を満たすように，該当するすべての τ_{12} を更新する．

   (c) $1 \in \{i_1, \cdots, i_p\}$ かつ $2 \in \{i_1, \cdots, i_p\}$ の場合：まず，r_{12} において，$\tau_{12} = \tau[R_1 \cup Y]$ を満たすすべての組 τ_{12} を更新し，更に，上記 (a) と (b) をともに適用する．

□

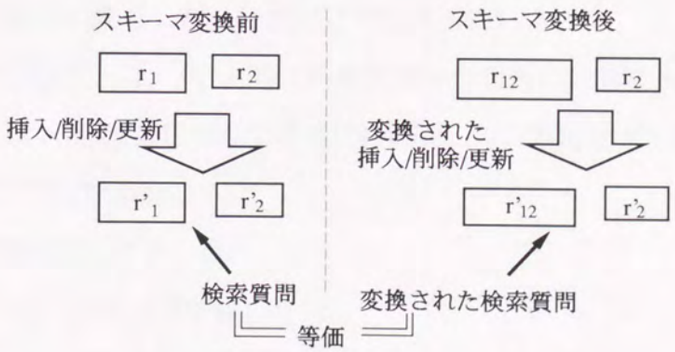


図 2.4 複写前後における検索結果の等価性

2.3.6 不整合を生ぜず，かつ，検索結果が等価になるための条件

2つの関係スキーマに対して複写を適用したときに，新しいデータベーススキーマにおいて変更質問を行うと，不整合を生じる可能性がある．しかし，定型業務のすべての変更質問がある条件を満たせば，不整合を生じないで，かつ，検索質問の結果も変換前の検索質問の結果と等しくなる（図2.4参照）．次の定理は，複写操作を適用してできた新しいデータベーススキーマにおいて，変換された質問が新たに不整合を生ぜず，かつ検索質問の結果が変換前と変換後で等価であるための必要十分条件を与える．

[定理 1] 関係スキーマ $\langle R_1, F_1\rangle, \langle R_2, F_2\rangle$ に対して，F_2に属する関数従属 $X \rightarrow Y$ を $\langle R_1, F_1\rangle$ へ複写して作成されたデータベーススキーマを **Rt** とする．**Rt** において，変換された変更質問がどのデータベースに対しても新たに不整合を生ぜず，かつ，検索質問の結果が変換前と変換後で等価であるための必要十分条件は，$\langle R_1, F_1\rangle$ または $\langle R_2, F_2\rangle$ を使うすべての質問が次の条件1から条件4を満たすことである．

r_1,r_2をそれぞれ，$\langle R_1, F_1\rangle$,$\langle R_2, F_2\rangle$ の上の基底関係とする．(r_1と r_2以外の基底関係については，不整合に関与しないのでここでは考えない．)

条件 1: r_1に対する挿入は，$R_1 \cup R_2$上の属性値を含むものに限られる．すなわち，r_1に対する挿入は r_1, r_2両方に対して同時に行われ，r_1のみに対する挿入はない．

条件 2: r_2に対して削除がない．

条件 3: r_1の X属性値は更新しない．

条件 4: r_2の $X \cup Y$ 属性値は更新しない．

(証明) $\langle R_1 \cup Y, F_1 \cup \{X \rightarrow Y\}\rangle$ を複写によって得られる新しい関係スキーマとし，その上の基底関係を r_{12}とする．

必要性：条件1，条件2，条件3は，補題2で述べたように，検索結果が等価であるための必要条件である．条件4が不整合を生じないための必要条件であることを以下に示す．

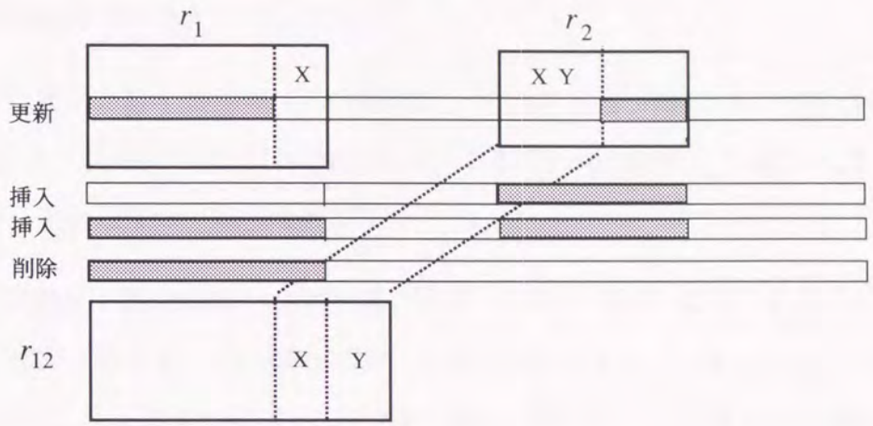


図 2.5 条件 1〜4 を満たす変更操作

$\langle R_1, F_1\rangle$ または $\langle R_2, F_2\rangle$ を使うある質問に対して，条件 4が成立しないと仮定する．このとき，r_{12}で不整合が新たに生じる例が存在することを証明する．r_2に属するある組の $X \cup Y$属性値に更新がある．例として，r_1, r_2がともに空の状態で，$\tau_1[R_1] \neq \tau_2[R_1]$ かつ$\tau_1[R_2] = \tau_2[R_2]$ を満足する $R_1 \cup R_2$上の２つの組τ_1, τ_2を挿入後，r_2に挿入された組$\tau_1[R_2]$ の $X \cup Y$属性値を更新する操作で得られるデータベースを考える．τ_1, τ_2の挿入後，$\tau_1[R_2] = \tau_2[R_2]$ より，$r_2 = \{\tau_1[R_2]\}$ が成立する．一方，r_{12}には２つの組が存在する．従って，r_2の組$\tau_1[R_2]$ の $X \cup Y$属性値の更新は，r_{12}に属する２つの組を更新する操作に変換され，更新不整合を生じる．

不整合を生じないことの十分性：$\langle R_1, F_1\rangle$ または $\langle R_2, F_2\rangle$ を使うすべての質問は条件 1〜条件 4を満たすと仮定する．このとき，r_1またはr_2を変更する操作は次の５通りに限られるので，それぞれの操作について新たな不整合を生じないことを証明する．

1. $R_1 \cup R_2$属性値をもつ組τの挿入：定義 3から，r_{12}に組$\tau[R_1 \cup Y]$ を挿入し，r_2に$\tau[R_2]$ を挿入する操作に変換されるので挿入不整合は生じない．

2. R_1属性値を持たずR_2属性値を持つ組τの挿入：r_2への$\tau[R_2]$ の挿入に変換され，新たな挿入不整合を生じない．

3. r_1に属する組τ_1 の削除：この削除は，$\tau_{12}[R_1] = \tau_1$ を満たすすべての組τ_{12} をr_{12} から削除する操作に変換される．ここで，τ_{12} の $X \cup Y$ 値は r_2 に残るので，新たな削除不整合は生じない．

4. r_1に属する組τ_1の $R_1 - X$属性値の更新：条件1〜条件4より，$\pi_X(r_1) \subseteq \pi_X(r_2)$ が成立するので，r_{12} の組の数は r_1 より少なくない．一方，$X \rightarrow Y$があることから，r_{12} の組の数は r_1 より多くない．従って，r_1 のすべての組τ に対して，それぞれ r_{12}に$\tau_{12}[R_1] = \tau_1$ を満たす組τ_{12} がちょうど一つ存在する．従って，この更新は r_{12} に属する$\tau_{12}[R_1] = \tau_1$ を満たす一つの組τ_{12} の更新に対応づけられ，新たな更新不整合は生じない．

5. r_2に属する組τ_2の $R_2 - (X \cup Y)$ 属性値の更新：r_{12}を変更せずに，r_2に属するτ_2だけを更新する操作に変換されるので，新たな更新不整合は生じない．

検索結果が等価であることの十分性:　次に，与えられたデータベーススキーマ **R** 上のデータベースに対する検索質問 Q_{ret}の結果は，複写操作により作られた新しいデータベーススキーマ **Rt** 上のデータベースに対する検索質問 $T(Q_{ret})$ の結果と等価であることを示す．ここで，$T(Q_{ret})$ は Q_{ret}を定義 2 に従って変換した質問である．r_1 または r_2 を使う変更質問を Q とし，Q を定義 3に従って変換した質問を，$T(Q)$ とする．また，Q の結果得られる関係を r_1', r_2'とし，$T(Q)$ の結果を r_{12}', r_2'とする．

定義 2の検索質問 Q_{ret}の変換では，$r_1 \bowtie \pi_{X \cup Y}(r_2)$ を r_{12}に置換し，この結合以外の r_1を$\pi_{R_1}(r_{12})$ に置換し，r_2はそのまま残す．従って，等価な検索結果を得るためには，次の (a)(b) が共に成立することを証明すれば良い．

(a) **R** における $\langle R_2, F_2 \rangle$ 上の基底関係と **Rt** における $\langle R_2, F_2 \rangle$ 上の基底関係が等しい．

(b) もし次の2つの式

$$r_{12} = r_1 \bowtie \pi_{X \cup Y}(r_2) \tag{2.1}$$

$$\pi_{R_1}(r_{12}) = r_1 \tag{2.2}$$

が成立するならば，変更質問の結果，

$$r'_{12} = r'_1 \bowtie \pi_{X \cup Y}(r'_2) \tag{2.5}$$

$$\pi_{R_1}(r'_{12}) = r'_1 \tag{2.6}$$

が成立する．

まず，(a) については，定義 3より，**R**, **Rt** のいずれのデータベーススキーマにおいても，r_2に対する操作は同じなので，r_2は **R** においても **Rt** においても同じ基底関係となる．

次に，(b) について，式 (2.1)(2.2) が成立すると仮定する．2通りの場合に分けて考える．

1. r_{12}が空の場合：

   式 (2.2) より r_1も空である．従って，Q は条件 1〜条件 4 により，$R_1 \cup R_2$属性値を持つ組τ の挿入，R_1属性値をもたず R_2属性値をもつ組の挿入および r_2に属する組の $R_2 - (X \cup Y)$ 属性値の更新の 3 種類だけである．τ の挿入は，r_{12} に対して組$\tau[R_1 \cup Y]$ を挿入する操作に変換されるので，式 (2.5)(2.6) が成立する．後者 2 つの操作に対しては，r_1が空なので明らかに式 (2.5)(2.6) が成立する．

2. r_{12}が空でない場合：

   (a) Q が挿入の場合は，$R_1 \cup R_2$属性値 をもつ組の挿入と，R_1属性値をもたず R_2属性値をもつ組の挿入の 2 種類だけである．$R_1 \cup R_2$属性値をもつ組τ の挿入は，r_{12}に$\tau[R_1 \cup Y]$ を挿入する操作に変換されるので，明らかに式

(2.6) が成立する．さらに$\pi_X(r_1) \subseteq \pi_X(r_2)$ が成り立つことおよび $X \to Y$ があることから，$\pi_X(r'_1) \subseteq \pi_X(r'_2)$ が成り立ち，式 (2.5) も成立する．

R_1属性値をもたず R_2属性値をもつ組τの挿入の場合は，r_1および r_{12}には変更がないので，式 (2.6) が成立する．また，$\pi_X(r'_1) \subseteq \pi_X(r'_2)$ が成り立つことから，式 (2.5) も成立する．

(b) Q が挿入以外の場合は，r_2に属する組の $R_2 - (X \cup Y)$ 属性値の更新，r_1に属する組の削除，および r_1に属する組の $R_1 - X$属性値の更新の 3 種類だけである．$R_2 - (X \cup Y)$ 属性値の更新は式 (2.5)(2.6) に影響を与えない．また，不整合を生じないことの十分性の証明で示したように，r_1の全ての組τ_1に対してそれぞれ r_{12}に$\tau_{12}[R_1] = \tau_1$を満たす組τ_{12}がちょうど一つ存在する．従って，後者 2 つの操作は，$\tau_{12}[R_1] = \tau_1$を満たすただ一つの組τ_{12}を更新または削除する操作に変換されるので，式 (2.5)(2.6) が成立する． □

2.4 関係スキーマの併合

2.4.1 併合の定義

$\langle R_j, F_j \rangle$, $\langle R_k, F_k \rangle$ において，F_kに属する関数従属を次々と $\langle R_j, F_j \rangle$ に複写する操作を繰り返すことにより，$R_j \supset R_k$となる場合がある．次に定義する併合は，このような場合に，さらに F_kに属する関数従属をすべて $\langle R_j, F_j \rangle$ に取り込み，$\langle R_k, F_k \rangle$ を消去する操作に対応する．

[定義 4] データベーススキーマ $\mathbf{R}$ に属する 2 つの関係スキーマ $\langle R_j, F_j \rangle$, $\langle R_k, F_k \rangle$ に対して，関数従属 $R_j \cap R_k \to R_k$ が F_k^+に存在するとする．このとき，この 2 つの関係スキーマを，新しい 1 つの関係スキーマ $\langle R_j \cup R_k, F_j \cup F_k \rangle$ で置換する操作を併合と定義する． □

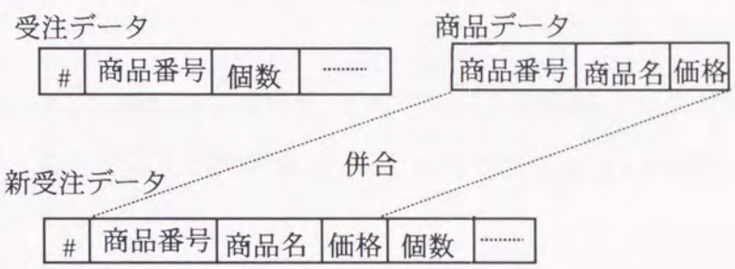


図 2.6 スキーマ併合の例

[例 3] 図2.6に示す3NF のスキーマが与えられるものとする．ここでは，例2と同様に，関係スキーマの属性名だけを [　　] 内に示す．新受注データは，受注データと商品データのスキーマを複写して出来た関係スキーマであり，二つのスキーマの属性と FD の和集合で構成される．受注データと商品データの両スキーマは消去する． □

以下，前章と同様に，関係スキーマ $\langle R_j, F_j \rangle, \langle R_k, F_k \rangle$ をそれぞれ $\langle R_1, F_1 \rangle, \langle R_2, F_2 \rangle$ とする．

2.4.2 検索質問の変換定義

データベーススキーマ **R** に属する 2 つの関係スキーマ $\langle R_1, F_1 \rangle, \langle R_2, F_2 \rangle$ を併合して作成したデータベーススキーマを **Rt** とする．また，**R** 上のデータベースを **D** : $r_1, r_2, r_3, \cdots, r_n$とし，**Rt** 上のデータベースを **Dt** : $r_{12}, r_3, \cdots, r_n$とする．ここで，r_1, r_2, r_{12}は，それぞれ関係スキーマ $\langle R_1, F_1 \rangle, \langle R_2, F_2 \rangle, \langle R_1 \cup R_2, F_1 \cup F_2 \rangle$ 上の基底関係とする．

併合の目的は，$R_1 \cup R_2$に対する検索が多いので，r_1と r_2のかわりに $r_1 \bowtie r_2$を r_{12}として記憶し，検索効率を上げることである．複写に伴って，定義 5に示すように，検索質問を変換する必要がある．

[定義 5] 併合に伴って，$\langle R_1, F_1\rangle$ または $\langle R_2, F_2\rangle$ を使う検索質問の操作を次のように変換する．

r_1と r_2を結合して得られる関係 $r_1 \bowtie r_2$に対する検索は，r_{12}に対する検索に変換する．$r_1 \bowtie r_2$以外に対する検索は，r_1,r_2をそれぞれ$\pi_{R_1}(r_{12})$, $\pi_{R_2}(r_{12})$ で置き換える． □

2.4.3 検索結果が等価であるための必要十分条件

定義5より明らかなように，併合後のデータベースに対する検索質問の結果が併合前のデータベースに対する検索結果と等価であるための必要十分条件は，次の 3 つの式がともに成立することである．

$$r_{12} = r_1 \bowtie r_2 \tag{2.7}$$

$$r_1 = \pi_{R_1}(r_{12}) \tag{2.8}$$

$$r_2 = \pi_{R_2}(r_{12}) \tag{2.9}$$

式 (2.7)(2.8)(2.9) から式 (2.10)(2.11) が導ける．

$$r_1 = \pi_{R_1}(r_1 \bowtie r_2) \tag{2.10}$$

$$r_2 = \pi_{R_2}(r_1 \bowtie r_2) \tag{2.11}$$

[補題 3] 併合後のデータベースにおいても，検索結果が等価となるための必要条件は，

$$\pi_{R_1 \cap R_2}(r_1) = \pi_{R_1 \cap R_2}(r_2) \tag{2.12}$$

が成立することである．

（証明）式 (2.12) が成立しないと仮定する．$\tau[R_1 \cap R_2] \notin \pi_{R_1 \cap R_2}(r_2)$ であるような組 τが r_1に存在する場合には，式 (2.10) が成立しない．また，$\tau[R_1 \cap R_2] \notin \pi_{R_1 \cap R_2}(r_1)$ であるような組τが r_2に存在する場合には，式 (2.11) が成立しない． □

2.4.4 等価が保証されるための変更質問の条件

併合前のデータベース D に対する検索質問の結果と等価な結果を，変換された検索質問が複写後のデータベース Dt から得るためには，D に対する変更質問がある条件を満たさなければならない．すなわち，D において，ある変更質問の結果，基底関係 r_1, r_2 が r_1', r_2' になったとすると，$\pi_{R_1 \cap R_2}(r_1') = \pi_{R_1 \cap R_2}(r_2')$ が必ず成り立たねばならない．このための必要条件を次の補題で示す．

[補題 4] 補題3の式 (2.12) が成立するための必要条件は，データベーススキーマ R 上のどのデータベース D に対しても，D に対するすべての変更質問が以下の 3 つの条件をすべて満足することである．

条件 5: r_1またはr_2に対する挿入は，$R_1 \cup R_2$上の属性値を含むものに限られる．すなわち，挿入は r_1, r_2両方に対して同時に行われ，r_1または r_2のみに対する挿入はない．

条件 6: r_1, r_2のいずれに対しても削除がない．

条件 7: r_1の $R_1 \cap R_2$属性値は更新しない．

(証明) 条件が 1 つでも成立しないとき，式 (2.12) が成立しないことを証明する．

条件 5が成立しない場合：例として，r_1, r_2が空の場合に，R_1上の組を挿入して得られるデータベースを考えると，式 (2.12) は成立しない．

条件 6が成立しない場合：例として，r_1, r_2が空のときに $R_1 \cup R_2$上の組を挿入すると，r_1, r_2にはそれぞれ 1 つだけ組が挿入される．その直後に r_1あるいは r_2の組の削除を行って得られるデータベースを考えると，式 (2.12) は成立しない．

条件 7が成立しない場合：例として，r_1, r_2が空のときに $R_1 \cup R_2$上の組を挿入すると，r_1, r_2にはそれぞれ 1 つだけ組が挿入される．その直後に r_1の組の $R_1 \cap R_2$属性値を更新して得られるデータベースを考えると，式 (2.12) は成立しない． □

2.4.5 変更質問の変換定義

複写の場合と同様に，データベーススキーマ **R** における変更質問を新しいデータベーススキーマ **Rt** における質問に変換する必要がある．スキーマを変換しても等価な検索結果が得られることが保証されるような変更質問のクラスに対して考えればよいため，定義 6では，補題 4の条件 5,6,7がすべて成立する場合についてのみ定義する．

[定義 6] 関係スキーマ $\langle R_1, F_1\rangle, \langle R_2, F_2\rangle$ に対して，その上の基底関係をそれぞれ r_1, r_2とする．また，$\langle R_1 \cup R_2, F_1 \cup F_2\rangle$ 上の基底関係を r_{12}とする．併合に伴って，$\langle R_1, F_1\rangle$ または $\langle R_2, F_2\rangle$ を使う変更質問を次のように変換する．

1. 挿入：データベース $\mathbf{D} : r_1, \cdots, r_n$ に対する $R_{i_1} \cup \cdots \cup R_{i_p}$上の組$\tau$の挿入は，$1 \in \{i_1, \cdots, i_p\}$ かつ $2 \in \{i_1, \cdots, i_p\}$ の場合だけであり，組$\tau[R_1 \cup R_2]$ を r_{12} に挿入する操作に変換する．

2. 更新：データベース $\mathbf{D} : r_1, \cdots, r_n$ に対する $R_{i_1} \cup \cdots \cup R_{i_p}$上の組$\tau$の更新を考える．

   (a) $1 \in \{i_1, \cdots, i_p\}$ かつ $2 \notin \{i_1, \cdots, i_p\}$ の場合：r_1に対する更新は，r_{12}において$\tau_{12}[R_1] = \tau[R_1]$ を満たすすべての組τ_{12} を更新する操作に変換する．(r_1, r_2, r_{12}に対する更新の結果を r_1', r_2', r_{12}'とするとき，$r_1' = \pi_{R_1}[r_{12}']$ が成り立つように，該当するすべての組を更新する．)

   (b) $1 \notin \{i_1, \cdots, i_p\}$ かつ $2 \in \{i_1, \cdots, i_p\}$ の場合：r_2に対する更新は，r_{12}において$\tau_{12}[R_2] = \tau[R_2]$ を満たすすべての組τ_{12} を更新する操作に変換する．($r_2' = \pi_{R_2}[r_{12}']$ が成り立つように，該当するすべての組を更新する．)

   (c) $1 \in \{i_1, \cdots, i_p\}$ かつ $2 \in \{i_1, \cdots, i_p\}$ の場合：まず，r_{12}において，$\tau_{12} = \tau[R_1 \cup R_2]$ を満たすすべての組τ_{12}を更新し，更に，上記 (a) と (b) をともに適用する． □

2.4.6 不整合を生ぜず，かつ，検索結果が等価になるための条件

併合を適用した後で，新しいデータベーススキーマにおいて変更質問を行うと，不整合を生じる可能性がある．しかし，定型業務のすべての変更質問がある条件を満たせば，不整合を生じないで，かつ，検索質問の結果も変換前の検索質問の結果と等しくなる（図2.7参照）．次の定理は，併合操作を適用してできた新しいデータベーススキーマにおいて，変換された変更質問が新たに不整合を生ぜず，かつ検索結果が変換前と変換後で等価であるための必要十分条件を与える．

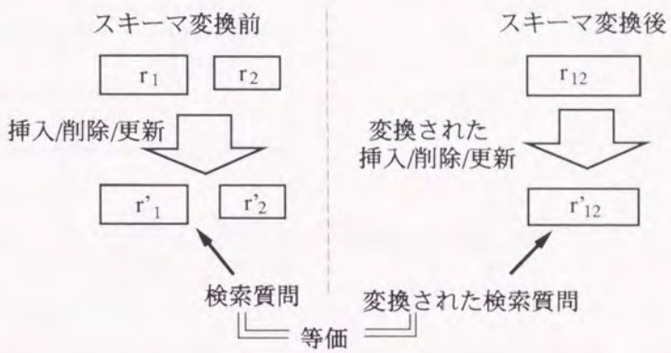


図 2.7 併合前後における検索結果の等価性

[定理 2] 関係スキーマ $\langle R_1, F_1 \rangle$ と $\langle R_2, F_2 \rangle$ を併合して作成されたデータベーススキーマにおいて，変換された変更質問がどのデータベースに対しても新たに不整合を生ぜず，かつ，検索質問の結果が併合前のデータベースに対する検索結果と等価であるための必要十分条件は，$\langle R_1, F_1 \rangle$ または $\langle R_2, F_2 \rangle$ を使うすべての質問が次の条件5〜条件8をすべて満たすことである．r_1, r_2をそれぞれ，$\langle R_1, F_1 \rangle$, $\langle R_2, F_2 \rangle$ の上の基底関係とする．

条件 5: r_1またはr_2に対する挿入は，$R_1 \cup R_2$上の組を含む．すなわち，挿入はr_1, r_2両方に対して同時に行われ，r_1 またはr_2のみに対する挿入はない．

条件 6: r_1, r_2のいずれに対しても削除がない．

条件 7: r_1の $R_1 \cap R_2$属性値は更新しない.

条件 8: r_2の R_2属性値は更新しない.

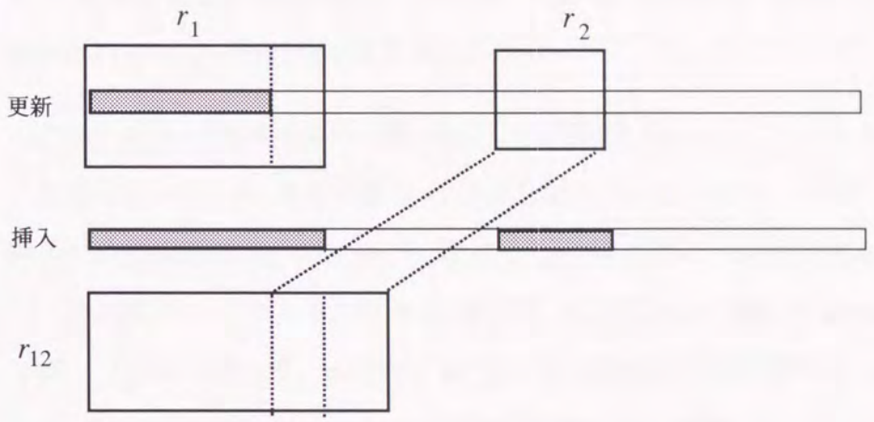


図 2.8 条件 5〜8 を満たす変更操作

(証明) $\langle R_1 \cup R_2, F_1 \cup F_2 \rangle$ を併合によって得られる新しい関係スキーマとし，その上の基底関係を r_{12}とする.

必要性： $\langle R_1, F_1 \rangle$ または $\langle R_2, F_2 \rangle$ を使う変更質問が，条件 5，条件 6，条件 7のいずれかを満たさなければ，検索結果が等価でなくなることを補題 4で証明した.

条件 8を満たさない場合に，基底関係 r_{12}で不整合が新たに生じる例が存在することを証明する. 仮定より，r_2に属するある組の R_2属性値の更新がある. 例として，r_1, r_2がともに空の状態で，$\tau_1[R_1] \neq \tau_2[R_1]$ かつ$\tau_1[R_2] = \tau_2[R_2]$ を満足する $R_1 \cup R_2$ 上の 2 つの組τ_1, τ_2を r_1, r_2へ挿入した後，r_2に属する組$\tau_1[R_2]$ の R_2属性値を更新する操作で得られるデータベースを考える. τ_1, τ_2の挿入後は，$\tau_1[R_2] = \tau_2[R_2]$ より $r_2 = \{\tau_1[R_2]\}$ がなりたつ. 一方，r_{12}には 2 つの組が存在する. 従って，組$\tau_1[R_2]$ の更新は，$\tau_{12}[R_2] = \tau_1[R_2]$ を満足する r_{12} の 2 つの組の更新操作に変換され，更新不整合が生じる.

不整合が生じないことの十分性：　$\langle R_1, F_1\rangle$ または $\langle R_2, F_2\rangle$ を使うすべての変更質問は条件5〜条件8を満たすと仮定する．このとき，r_1 または r_2 を変更する操作は次の2通りだけであり，ともに不整合を生じないことを証明する．

1. $R_1 \cup R_2$ 属性値を含む組の挿入：これは，r_{12} へ組 $\tau[R_1 \cup R_2]$ を挿入する操作に変換されるので，新たな不整合は生じない．

2. r_1 に対する $R_1 - R_2$ 属性値の更新：条件5〜条件8より，$\pi_{R_1 \cap R_2}(r_1) = \pi_{R_1 \cap R_2}(r_2)$ となる．従って，r_{12} の組の数は r_1 の組の数より少なくない．一方，定義6の挿入の変換規則と $R_1 \cap R_2 \rightarrow R_2$ より，r_{12} の組の数は r_1 の組の数より多くない．それゆえ，r_1 の各々の組 τ に対して，$\tau_{12}[R_1] = \tau$ を満たす組 τ_{12} が r_{12} に丁度一つある．従って，r_1 に対する $R_1 - R_2$ 属性値の更新操作は，r_{12} に属する $\tau_{12}[R_1] = \tau$ を満たす1つの組 τ_{12} の更新操作に対応づけられ，r_{12} において新たな不整合は生じない．

検索結果が等価であることの十分性：

r_1 または r_2 を使う変更質問を Q とし，Q を定義6に従って変換した質問を $T(Q)$ とする．Q の結果得られる関係を r_1', r_2' とし，$T(Q)$ の結果得られる関係を r_{12}' とする．また，与えられたデータベーススキーマ上のデータベースに対する検索質問を Q_{ret} とする．定義6により，Q_{ret} 中の $r_1 \bowtie r_2$ を r_{12} で，r_1 を $\pi_{R_1}(r_{12})$ で，r_2 を $\pi_{R_2}(r_{12})$ でそれぞれ置換するので，検索結果が等価であるためには，もし次の3つの式

$$r_{12} = r_1 \bowtie r_2 \tag{2.7}$$

$$\pi_{R_1}(r_{12}) = r_1 \tag{2.8}$$

$$\pi_{R_2}(r_{12}) = r_2 \tag{2.9}$$

が成立するならば，変更質問の結果，

$$r_{12}' = r_1' \bowtie r_2' \tag{2.13}$$

$$\pi_{R_1}(r_{12}') = r_1' \tag{2.14}$$

$$\pi_{R_2}(r_{12}') = r_2' \tag{2.15}$$

が成立することを証明すればよい.

式 (2.7) (2.8) (2.9) が成立すると仮定し，次の 2 通りの場合に分けて考える.

r_{12}が空の場合：仮定より，r_1, r_2ともに空となる．空であることおよび条件 5より，データベースを変更する質問 Q は $R_1 \cup R_2$上の組τの挿入だけである．この挿入は，r_{12}に対して組$\tau[R_1 \cup R_2]$ を挿入する操作に変換されるので，式 (2.13) (2.14) (2.15) が成立する.

r_{12}が空でない場合：仮定より，r_1, r_2ともに空でない．Q が挿入操作の場合は，$R_1 \cup R_2$属性を持つ組τの挿入だけであり，これはr_{12}に対して組$\tau[R_1 \cup R_2]$ を挿入する操作に変換されるので，式 (2.14) (2.15) が成立する．また，$\pi_{R_1 \cap R_2}(r_1) = \pi_{R_1 \cap R_2}(r_2)$ であることから，組$\tau[R_1 \cup R_2]$ の挿入後，式 (2.13) が成立する．Q が挿入以外の場合の変更操作は，r_1に属するある組の $R_1 - R_2$ 属性値の更新のみである．r_1の全ての組τ_1に対してそれぞれ r_{12}に$\tau_{12}[R_1] = \tau_1$を満たす組がちょうど一つ存在することから，この更新はr_{12}に属するある一つの組だけを更新する操作に変換されるので，式 (2.13) (2.14) (2.15) が成立する. □

2.5 定理の必要十分条件を満たさない場合の適用可能性

定理 1, 2 で示した条件 1〜8 は，データベースの内容とは独立な条件である．しかし，与えられたデータベーススキーマと質問が条件 1〜8 を満たさない場合でも，変更操作の直前でデータベースの内容に関してある条件が成立すれば，複写／併合を行うことができ，かつ，与えられた質問は不整合を生じない（等価な検索結果が得られるような）質問に変換できることがある.（簡単のため，以下，複写／併合が可能であるという.） また，不整合は個々の変更操作に対して定義されていたが，複数の操作を一括して実行するような SQL のトリガー機構などを用いれば，その処理中には検索が行われないようにできるので，一括操作の中では不整合とみなさなくてもよい場合がある．以下では，上記のことを考慮して，複写が可能であるため

の十分条件および対処の方法を簡単に述べる.(併合についてもほぼ同様に考えることができる.)

まず，複写の条件 1 に関しては，条件 1 が成立しないとき，挿入する組τ は R_1 属性値を持つが R_2属性値を持たない場合がある．この場合においても，このようなすべてのτの挿入の直前で $\tau[X] \in \pi_X(r_2)$ が成立すれば，複写が可能である．なぜなら，図 2.9に示すように，τ の挿入は，定義から，r_{12} に新しい組 τ_{12} を挿入する操作に変換されるが，もし $\tau[X] \in \pi_X(r_2)$ ならば，$\tau_{12}[Y]$ の値は r_2 から求められるので，挿入不整合が生じないからである．

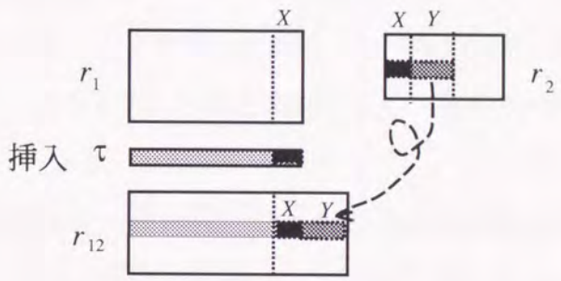


図 2.9 r_1への挿入操作の変換

条件 2 が成立しない場合には，r_2の組の削除が存在する．削除する組をτ_2とすると，このようなすべての削除の直前で $\tau_2[X] \notin \pi_X(r_1)$ が成立すれば，複写が可能である．これは，以下の理由による．τ_2の削除は，(1) r_2 からτ_2 を削除し，(2) r_{12} から $\tau_{12}[X] = \tau_2[X]$ を満たすすべての組 τ_{12} を削除する操作に変換するのが妥当である．このとき，もし $\tau_2[X] \notin \pi_X(r_1)$ ならば，$\tau_2[X] \notin \pi_X(r_{12})$ も成立するので，r_{12} から削除する組は存在せず，削除不整合は生じない．

条件 3 が成立しない場合には，r_1の X属性値の更新が存在する．r_1 の組τ_1の X属性値を$\tau'[X]$ に更新するとした時，このようなすべての更新の直前で $\tau'[X] \in \pi_X(r_2)$ が成立すれば，複写が可能である．これは以下の理由による．τ_1の更新は，$\tau_{12}[R_1] = \tau_1$ を満たす r_{12} の組 τ_{12} を更新する操作に変換されるが，そのような τ_{12} は r_{12} にた

だ一つしか存在しない．また，もし $\tau'[X] \in \pi_X(r_2)$ ならば，$\tau_{12}[Y]$ の値は一意に決定される．従って，更新不整合は生じない．

条件 4 が成立しない場合は，データベースの内容に関わりなく，SQL のトリガー機構を用いて以下のように対処できる．即ち，条件 4 が成立しない場合，r_2の $X \cup Y$ 属性値を更新する操作が存在する．τ_2をそのとき更新される組とする．定義 3に従って，τ_2の更新は，(1) r_2 において τ_2 を更新し，(2) r_{12} において$\tau_{12}[X \cup Y] = \tau_2[X \cup Y]$ を満たすすべての組τ_{12} を更新する操作に変換される．定義から，後者の操作は更新不整合である．しかし，「r_2の $X \cup Y$属性に対する更新処理が始まる」というイベントをトリガー処理の起動条件としてあらかじめ設定しておき，トリガー処理の中で複数の更新を一括して行うようにすればよい．(トリガー処理の中では，複数の更新は原子的（Atomic），すなわち，不可分な処理として扱えるので，更新不整合とは考えない．)

なお，トリガー機構の中で用いられるような排他制御機能を質問の中で用いることにより，条件 1〜3 が成立しないときでも，複写が可能であるための十分条件を拡張することは可能である．しかし，これらは個別のデータベース管理システムに依存するため，汎用的な必要十分条件を求めるのは困難である．

2.6 実システムでの適用例

A 社で実稼働している販売管理システムのデータベーススキーマを分析して，非正規化が行われていることに気づいたことが，筆者がこの研究を進める動機となったことは最初に述べた．本節では，システムの概要を紹介し，複写や併合がどの程度適用できるかを考察する．

2.6.1 システムの概要

システムでの業務の流れの概要を，図 2.10に示す．本節では，「商品販売在庫管

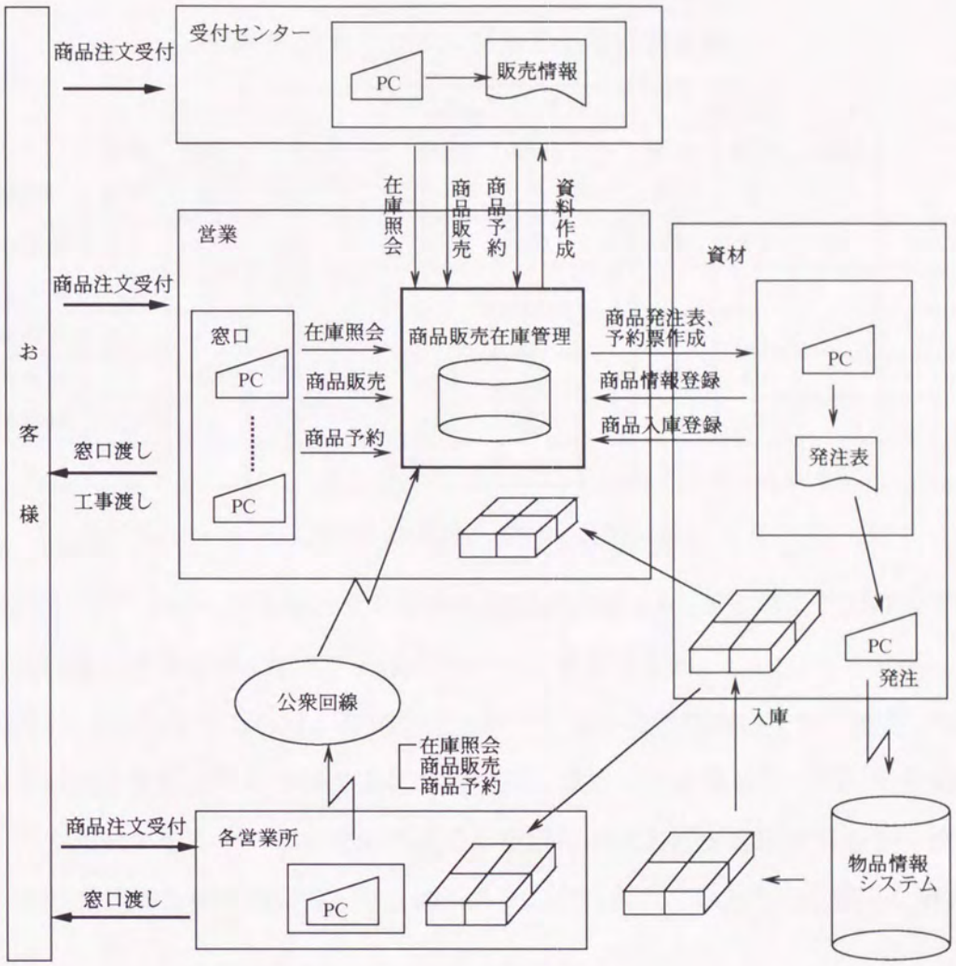


図 2.10 システムの業務の流れ

理」と記載されている部分について考察する．データベース・サーバは，パソコンで運用されており，販売窓口や地方営業所のパソコンから LAN や公衆回線を通して送られたトランザクションをさばいている．ソフトウェア構成は，以下の通りである．

- 使用 OS : MS-DOS Ver. 3.3
- LAN S/W : MS-Network StarLAN
- DBMS : R:BASE

表 2.1 業務とテーブルの処理の関連表

	在庫テーブル			…	商品テーブル		…	販売テーブル			…	…
業務名	商品番号	在庫数	販売総数	… …	商品コード	商品名	… …	販売番号	販売日	商品番号	… …	… …
販売登録	○	◎	◎	…	○	○	…	◎	◎	◎	…	…
商品入庫	○	◎	○	…								
在庫照会	○	○	○	…								
予約販売	○	◎	◎	…	○	○	…	◎	◎	◎	…	…
商品登録	◎	◎	◎	…	◎	◎						
……												

（○印：参照項目，◎印：変更項目）

また，データベーススキーマと業務の関連は，表 2.1に示したような形式で整理されている．表の中で，テーブル名はスキーマ名をとする．

実は，このシステムには，非正規化されているところを除いても，まだ 3NF になっていない関係スキーマがある．本節では，スキーマを整理し 3NF に正規化した上で，提案するスキーマの変形による非正規化がどの程度適用できるかを検討する[3]．非正規化する属性項目は，もとのシステムで行われていたものを第一に考える．

2.6.2 関数従属の複写の例

商品販売管理業務をとりあげる．２つの関係スキーマにおいて，ある関数従属を複写して，質問の結合処理を減らしている例を説明する．

この定型業務では，主に次の２つの関係スキーマが使われるものとする．なお，表現を簡単にするために，例の中では属性名だけを〈 〉内に書く．

関数従属は，「商品番号 → 在庫数, 販売総数, 商品コード, 色」と，「商品コード → 商品名, 定価, 記事」である．同じ商品コードでも，色によって商品番号が異なるの

[3]このシステムの内容に関して，A 社と内容非公開の約束をしているため，スキーマ構成，スキーマ名，属性名などは，実際のシステムのものから変更してある．

で，このようなスキーマとなっている．

- 在庫：〈 商品番号, 在庫数, 販売総数, 商品コード, 色 〉
- 商品：〈 商品コード, 商品名, 定価, 記事 〉

また，使われる質問は次のように固定されている．

1. 商品の販売時には必ず在庫の確認をするが，常に商品名も一緒に表示する（結合を含む検索質問）．
2. ある商品が売れたとき，在庫数を減らし，販売総数を増やす（更新質問）．
3. 商品を入庫するとき，在庫数を増やす（更新質問）．
4. 新商品の登録は，「在庫」と「商品」の２つのスキーマ上の基底関係に対して組を挿入する（挿入質問）．
5. 販売をやめた商品は，記事欄にその旨を書く（削除は無い）．

このように質問で使われるアクセス経路が固定しており，「商品コード→ 商品名」の関数従属が成り立つときに，「商品コード→ 商品名」を複写し次のような新しいスキーマを作って，「在庫」のスキーマを置き換える．また，もとの「商品」のスキーマはそのまま残す．

- 在庫管理：〈 商品番号, 在庫数, 販売総数, 商品コード, 色, 商品名 〉

在庫照会（検索）は，新しい「在庫管理」スキーマを使えば結合操作をしなくてよくなり，処理性能の向上を図ることができる．また，在庫数と販売個数の値の更新も，「在庫管理」スキーマを使って不整合を生ぜずに実行することができる．

2.6.3　スキーマ併合の例

商品販売管理業務で，２つの関係スキーマを一つに併合して，質問の結合処理を減らす例を説明する．

- 販売：〈 販売番号, 販売日, 商品番号, 売値, 個数, 顧客電話番号 〉
- 顧客：〈 顧客電話番号, 顧客名 〉

また，定型業務で使われる質問は次のように固定されている．

1. ある商品が売れたとき,「販売」∪「顧客」の上の１つの組τ を挿入する．すなわち,「販売」と「顧客」の２つのスキーマ上の基底関係に対して，それぞれτ[販売]，τ[顧客] を挿入する．
2. 「販売」のスキーマを使って検索処理をするときに，常に「顧客」の基底関係を結合して，検索結果に顧客名も併せて出力する．
3. 顧客電話番号と顧客名は通常更新しない．更新が必要な時には，該当の顧客データが複数組あれば全て更新を行う．

このように質問で使われるアクセス経路が固定しており,「顧客電話番号→顧客名」の関数従属が成り立つので，２つのスキーマを併合し次のような新しいスキーマを作る．

- 販売管理：〈 販売番号, 販売日, 商品番号, 売値, 個数, 顧客電話番号, 顧客名 〉

商品を販売したときの登録処理は，新しい「販売管理」スキーマを使って顧客名も一緒に挿入するように，業務プログラムを変換する．実際には，A 社では，この 2 つのスキーマが併合された形で運用されている．顧客電話番号と顧客名の更新が必要な時には，該当の顧客データが複数組あればその全てに対して更新を行うように，業務プログラムが作成されている．

データベースに登録されているデータの件数やトランザクションの頻度などの実データがないため，定量的な評価はできないが，検索処理は新しい「販売管理」スキーマを使うように変換することにより，結合操作をしなくてよくなり，検索処理の性能の向上を図ることができる．

2.7 性能について

データベース管理システムは，データベースバッファ，または，キャッシュと呼ばれる領域を仮想記憶上に準備して，入出力の高速化を行えるようにしている．この領域のサイズはデータベース管理者が設定する．このサイズが大きければ，最初のアクセスの時にディスクから読み込んだ基底関係の組を一時的に保管できる可能性が高いので，二回目からの検索が速くなり，さらにバッファ上で結合処理ができる可能性があることから結合を伴う検索処理が高速に行えるなどの利点がある．

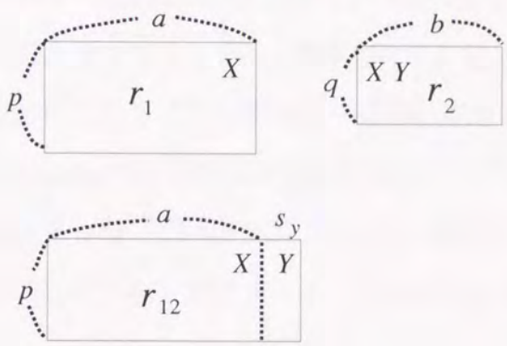


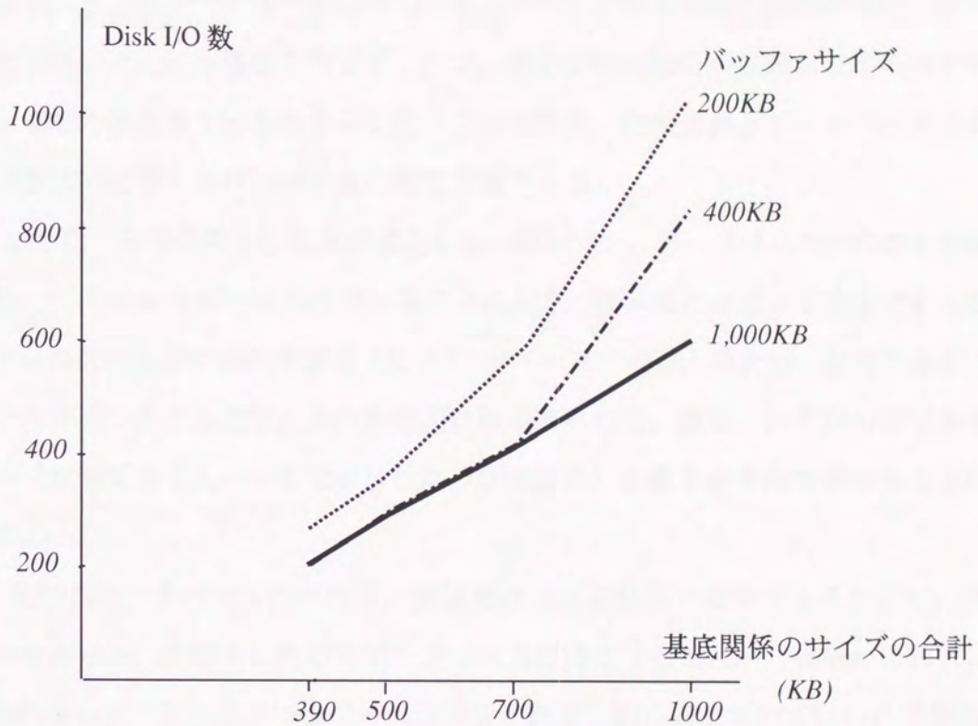
図 2.11 複写を適用後の基底関係のサイズの変化

複写を適用した場合，基底関係のサイズが変わる．r_1の一つの組のサイズを a バイト，組の数を p とする．同様に，r_2 のサイズを b, q とする．また，データベース管理システムは，組を管理するために組ごとに識別子[4]を付けるので，そのサイズを c バイトとする．r_1, r_2 のサイズは，それぞれ，$p*(a+c), q*(b+c)$ と表せる．また，$X \rightarrow Y$を複写するとしたとき，Y のサイズを s_y バイトとすると，r_{12} のサイズは，$p*(a+b-s_y+c)$ バイトとなる．従って，データベースのページサイズを P とすると，基底関係をディスクから読み込むときの I/O 回数の節約分 N は，

[4]商用の関係データベース管理システムである ORACLE では，これを row-id と呼ぶ．row-id のサイズは 5 バイトである．

$\lceil p*(a+c)/P \rceil + \lceil q*(b+c)/P \rceil - \lceil p*(a+s_y+c)/P \rceil$ となる．$\lceil x \rceil$ は，x の小数点以下を繰り上げた整数を表す．繰り上げを無視して式を変形すると，$N = q*(b+c) - p*s_y$ が得られ，p の値（r_1の組の数）が大きいとI/O回数の節約効果が減ることになる．

データベースバッファのサイズが，r_1, r_2 のサイズの和より大きい場合には，結合処理がバッファ上で実行できるため，スキーマを併合した後では，メモリ上での $p*q$ 個の組の結合処理の時間と，N 回だけのディスク read の時間が節約できる．しかし，データベースバッファは他の質問にも用いられるので，実際には，r_1, r_2 のサイズの和より大きなサイズのバッファ領域が使えるとは限らない．この場合，複写を適用する前のデータベースでは，結合処理を行うためにキャッシュ上に基底関係を何度か読み直すことが必要になるため，p の値が大きいほどディスクI/O回数が増え，結合処理の時間が長くなる (図 2.7参照)．複写を適用した後のデータベースでは，結合処理が無いため r_{12}の組を順次読み込めばよく，バッファサイズが十分でなくても，検索性能は低下しない [Tsuji 91]．従って，複写または併合を適用して，あらかじめ基底関係を結合した形で蓄積すると，性能が向上する．



r_1 のサイズ： 35byte/組，10,000 組で固定

r_2 のサイズ： 35byte/組，1,000 組，5,000 組，10,000 組，20,000 組

実行文："SELECT $r_1.*, r_2.*$ FROM r_1, r_2 WHERE ..."

図 2.12 基底関係のサイズと，結合を含む検索の I/O 数との関連

（文献 [Tsuji 91] より改訂）

2.8 むすび

定型業務とそのデータベーススキーマが与えられたとき，検索の処理効率を改善するために，関係スキーマを変形する「複写」と「併合」の２つの操作を定義した．そして，どんなデータベースに対しても，スキーマの複写または併合後に，変換された質問が新たに不整合を生ぜず，かつ，検索質問の結果が変換前と変換後で等価であるための必要十分条件を示した．この条件は，定型業務とデータベーススキーマが形式的に与えられれば容易に判定可能である．

さらに，その必要十分条件が成立しない場合でも，データベースの内容を考慮したり，SQL のトリガー機能を用いることにより，複写または併合を適用できるための十分条件ならびに対処方法を は，データベースの内容，および，利用するデータベース管理システムで提供される機能に依存するため，複写／併合が可能であるための（本論文の 2.3，2.4節で示したような簡潔な）必要十分条件を求めることは困難である．

実際のデータベースにおいて，定型業務の検索処理を高速化するために，基底関係をあらかじめ結合した形でデータベースに持たせることが，従来からしばしば行われていた．これはデータベース設計者の経験と勘にもとづいて行われてきたが，本論文の結果から，結合しても良いかどうかは機械的に判定できるようになり，正しい設計が勘にたよらずに可能になる．

第 3 章

タンパク質のオブジェクト指向データベースのスキーマ設計

3.1 まえがき

分子生物学の分野では，個々の遺伝子の機能が何か，即ち，遺伝子から作られるタンパク質がどんな機能を持つかを調べることが重要である．タンパク質の機能は，立体構造と密接な関連をもつことがわかってきているので，遺伝子からデコードしたアミノ酸系列を，立体構造が解析されたタンパク質のアミノ酸系列と比較して，遺伝子の機能を推定する手法がよく用いられている．1970 年代後半には，タンパク質の構造データを自由に検索・参照したいという要求に応じて，タンパク質のデータバンクとして，Protein Data Bank (PDB) が米国 Brookhaven 国立研究所で作成された [Bern 77]．データは磁気テープで配布され，各地の計算機センターなどで 検索システムが構築されている．しかし，センター提供の検索システムでは，利用者が検索したい条件を表せない場合があり，利用者が自分で Fortran などを用いて検索プログラムを作成する必要があった．利用者にとってプログラムの作成は必ずしも容易ではないので，プログラムを作成することなく，自由な検索を行いたい

Structure relation

BRCODE	PNAME	PFUNCTION	PSOURCE	.....
4HHB	HEMOGLOBIN	TRANSPORT	HUMAN..	.....
2PAB	PREALBUMIN	TRANSPORT	HUMAN..	.....
...	...	...	...	...

Chain relation

BRCODE	CHAINID	NRES	...
4HHB	A	141	...
4HHB	B	146	...
4HHB	C	141	...
4HHB	D	146	...
2PAB	A	127	...
2PAB	B	127	...
...	...	...	...

Residue relation

BRCODE	RNAME	POSITION	CHAINID	...
...	...	...	...	...
4HHB	SER	3	A	...
4HHB	PRO	4	A	...
4HHB	ALA	5	A	...
...	...	...	...	...
4HHB	VAL	17	A	...
4HHB	GLY	18	A	...
...	...	...	...	...

図 3.1 BIPED のデータ表現

という要求がある.

一方, X 線による結晶解析技術や核磁気共鳴 (NMR) 技術による非結晶解析技術の進歩に伴い, 構造の解析されたタンパク質の数が最近急増している. また, PDB では, データは個々のタンパク質毎に Fortran カード形式のテキストファイルとして格納されているので, 情報を表現しにくく, データの収容効率も良くない. これらの背景から, データベース管理システムを用いたタンパク質データベースが試作されてきた.

関係データベース管理システムを用いたタンパク質データベースとして, BIPED (Birbeck Integrated Protein Engineering Database), SESAM, PACADE (Protein Atomic Coodinate Analyzer with Deductive Engine) などがある. BIPED では, データベース管理システムとして ORACLE を用い, 検索には SQL を用いる [Islam 89] .

データベーススキーマは，タンパク質の構造を反映させて，Structure, Chain, Residue, Atom などの 10 個の関係スキーマで構成されている．図 3.1に，一部の基底関係を示す．PACADE も SYBASE を用いてデータを蓄積しているが，演繹機能をもったプログラムモジュールと組み合わせて，超二次構造の検索なども出来るように工夫している [Kuha 91, Satou 94] ．一方，SESAM では，データベース管理システムとして SYBASE を用い，PDB だけでなく，SWISS-PROT などのタンパク質の系列のデータバンクからのデータを統合して蓄積している [Huys 91]．また，検索は基本的に SQL を用いる．

以上のように，関係データベースを用いると，SQL によって簡単な検索ができるという利点があるが，次のような欠点がある．

(1) 系列データを扱いにくい．アミノ酸系列を基底関係に記憶させるためには，個々のアミノ酸に順序番号を陽に指定する必要がある．

(2) キーや順序番号によってデータが冗長になる．さらに，系列を得るためには，基底関係を join する操作が必要となる．

(3) 計算を伴うような複雑な検索条件は，SQL で表現しにくい．

これらの問題は，オブジェクト指向データベースを用いることで，解決できる．筆者の研究グループでは，商用のオブジェクト指向データベース管理システム VERSANT を用いてタンパク質構造のデータベースを試作した．上記の問題点 (1) (2) は，個々のタンパク質を図 3.2に示すように，複合オブジェクトで表現することにより対応できる．即ち，鎖，2 次構造，アミノ酸，原子など，タンパク質を構成している要素をクラスとし，*is-part-of* 関係で各クラスを関係づけることによってタンパク質を複合オブジェクトで表現できる．従って，タンパク質の構造をデータベーススキーマに自然に反映させることができ，スキーマの設計が容易である．

問題点 (3) については，メソッドを用いることである程度まで解消できる．即ち，オブジェクト指向データベースでは，オブジェクトに対する処理をメソッドと

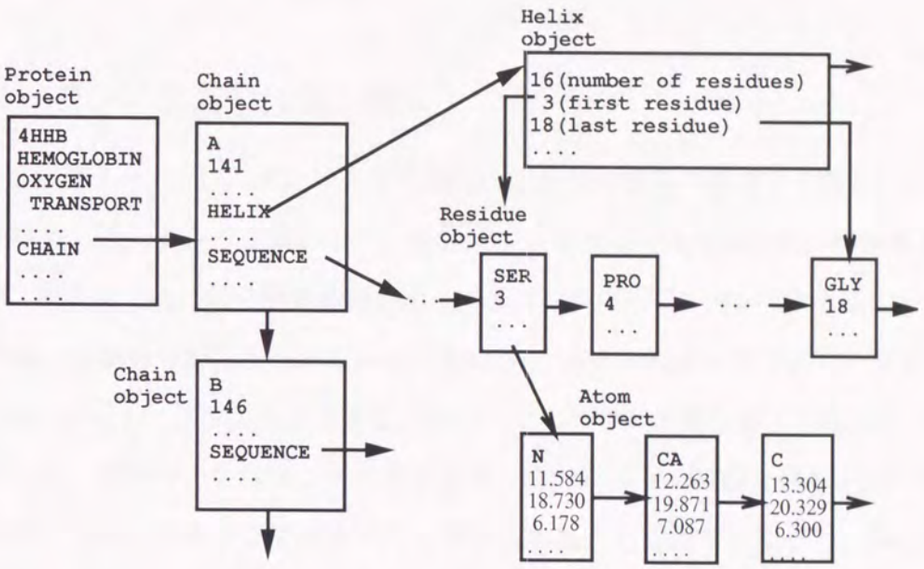


図 3.2 タンパク質の概念構造の表現

して記述できるので，データベース管理者が検索手段としてメソッドをあらかじめ用意しておくことにより，利用者はメソッドを用いて検索文を簡潔に記述することができる．詳細は3.3節で述べる．

上述のようにスキーマを設計しメソッドを実現すると，一応データベースとして使える状態になるが，そのままでは，クラス階層にうまく適合しないような検索質問に対する処理の効率が悪くなるという弱点がある．また，原子を表すオブジェクトの数が膨大となり，ディスク容量が増大するという問題点がある．これらの対応策について，3.4節で述べる．

さらに，利用者にとってできるだけ使いやすいようにするため，本データベース上で動作するグラフィカル・ユーザ・インタフェース (GUI) として，*Protein Browser*を作成した．GUI および，検索処理を効率化するための仮想クラスについても，3.4節で述べる．

最後に，更新について述べ，関連する研究との比較を行う．

3.2 データベーススキーマ

3.2.1 タンパク質の分類と構成

本データベースでは，タンパク質を機能の観点から酵素の種類で分類し，クラス階層を構成した．タンパク質のうち，酵素についてはほとんどの場合，酵素番号[1]が割り当てられている．この酵素番号によって $EC.1$ から $EC.6$ のいずれかのクラスに分類する．酵素番号が付けられていないものは $Other\_enzyme$ クラスに，また非酵素は Non_enzyme クラスに分類する．スキーマのクラス階層を図 3.3 に示す．楕円はクラスを，楕円中の名前はクラス名を表す．有向辺は $is\text{-}a$関連を表し 矢印の指すクラスは，矢印の出るクラスのスーパクラスであることを示す．従って，$Enzyme$, および $Other\_enzyme$ は $Protein$ からすべての属性とメソッドを継承する．同様に $EC.1$ から $EC.6$ ，および $Other\_enzyme$ の 7 つのクラスは $Enzyme$ からすべての属性とメソッドを継承する．

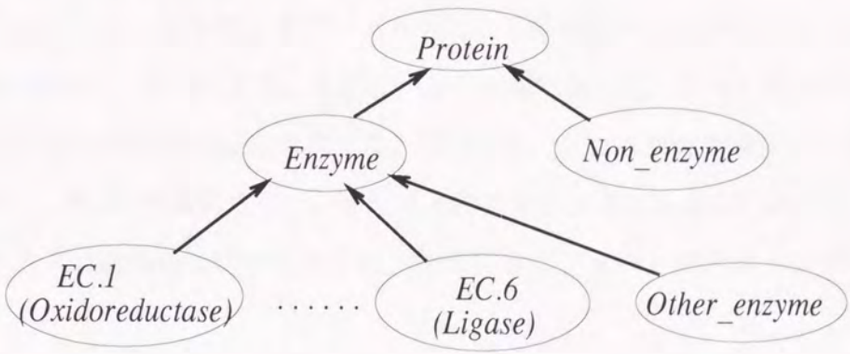


図 3.3 クラス階層 ($is\text{-}a$関連)

図 3.4は，$Protein$ クラスにおけるタンパク質の表現方法を示している．個々のタンパク質は，アミノ酸がつなぎ合わされてできた鎖 ($Chain$) と呼ばれる構造がい

[1]酵素を系統的に分類するための，4 組の数字からなる番号．

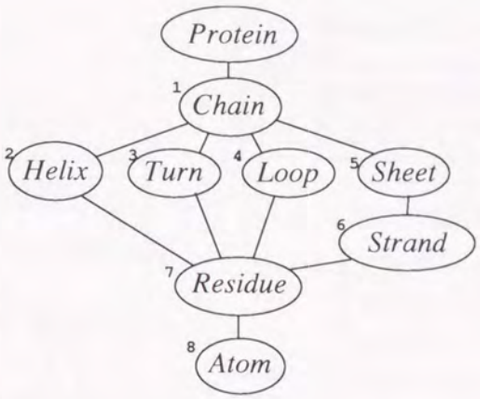


図 3.4 複合オブジェクトの構成 (*is-part-of* 関連)

くつか集まって構成されている．鎖は 4 種の 2 次構造，ヘリックス (*Helix*)，ターン (*Turn*)，ループ (*Loop*)，シート (*Sheet*) から構成され，さらに，2 次構造はアミノ酸残基 (*Residue*) の系列から，アミノ酸残基は複数の原子 (*Atom*) から構成される．これらの構成要素をそれぞれクラスとし，1 つのタンパク質をこれらのクラスのオブジェクトから成る複合オブジェクトとして構成することができる．これらのクラスは *is-part-of* 関連にある．上述のクラス階層に従って，すべてのタンパク質オブジェクトはこの表現方法を継承する．図 3.5は，タンパク質の複合オブジェクトを，著者 (*Author*) や雑誌名 (*Journal*) など関連するクラスを含めて表している．四角のボックスは個々のオブジェクトを，矢印はオブジェクト間のポインタを表す．

3.2.2 クラス属性

属性には 2 種類がある．一つは，PDB でのレコード（Fortran カード）を識別するキーワードで，もう一つは，REMARK レコードの中に書かれた英語文によるコメントから抽出したキーワードである．後者は，よく検索されるキーワードをあらかじめ属性として切り出しておき，検索の効率を高めるために用意するもので，解像度 (**Resolution**) などがこれに相当する．ここでは，特徴的な属性をいくつか紹

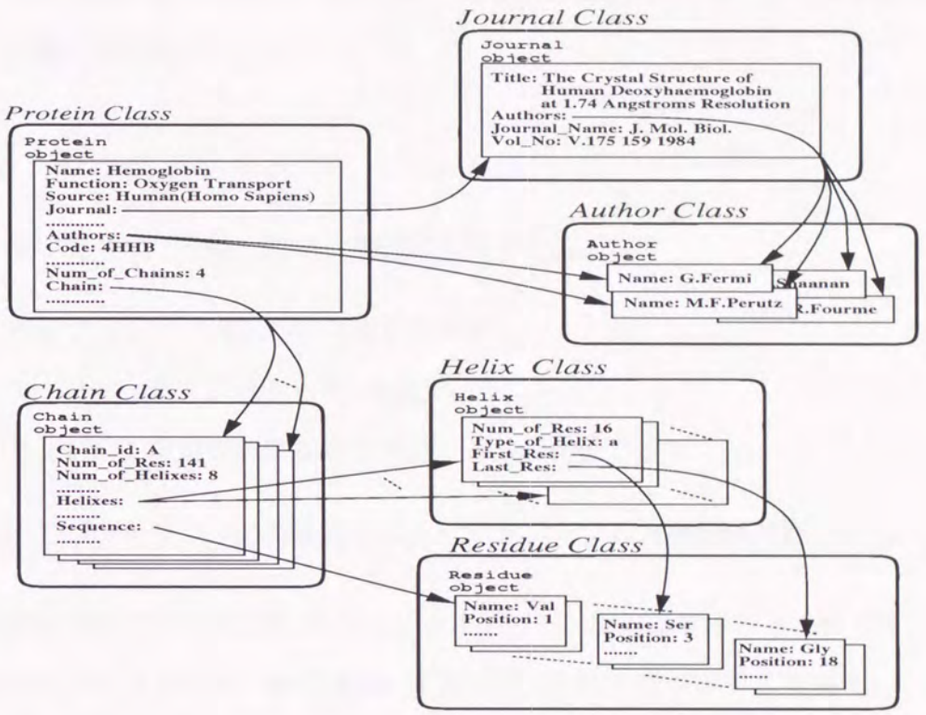


図 3.5 複合オブジェクトによるタンパク質の表現

介する.

1. *Protein* クラスは，24 コの属性を持つが，ここでは以下の 5 コを説明する.(詳細は付録に示す.)

   - **Name**: タンパク質の名称
   - **Functional_Classification**: タンパク質の機能名
   - **Author**: 著者オブジェクトへのポインタリスト
   - **Structure_Analysis_Method**: 構造を調べる時に用いた手法
   - **Resolution**: 構造を調べる時に用いた X 線の波長

2. *Enzyme* クラスは，*Protein* クラスの属性を継承する他，**EC_number** を追加属性として持つ.

3. *EC*.1 ～ *EC*.6 と *Other_enzyme* クラスは，*Enzyme* クラスの属性を継承する.

4. *Non_enzyme* クラスは，*Protein* クラスの属性を継承する．このクラスに固有の属性はない．

3.2.3 クラスメソッド

定義したメソッドは，次の 3 種類に分類できる．

1. タイプ A：文字列型の属性値を表示する．
2. タイプ B：オブジェクト ID を返す．
3. タイプ C：指定された条件を判定して，真偽値を返す．

本節では，いくつかの特徴的なメソッドを紹介する．(詳細は付録に示す．)

1. *Protein* クラスは，約 50 のメソッドを持つ．次に示すメソッドのうち，最初の 3 つはタイプ A，**get_chain** はタイプ B，残りはタイプ C である．

   - **display_all**: タンパク質の全ての文字列型の属性値を表示する．
   - **display_name**: タンパク質の名前を表示する．
   - **display_3D**: タンパク質の立体構造を表示する．
   - **get_chain**: タンパク質を構成するすべての鎖オブジェクトのオブジェクト ID のリスト返す．
   - **judge_sequence**: 指定されたアミノ酸系列パターンがタンパク質に含まれるかどうか判定し，真偽値を返す．パターンとしては，例えば，“Ala-{Gly, Ser}-?-Ala-∗-Gly” を与える．これは，最初のアミノ酸がアラニンで，次にグリシンかセリンが続き，何か 1 つのアミノ酸をはさんでアラニンが来て，さらに零個以上のアミノ酸をはさんでグリシンがあるという部分系列を表す．
   - **judge_structure**: 指定された 2 次構造のパターンがタンパク質に含まれるかどうか判定し，真偽値を返す．例えば，“bab” は，βシートにαヘリックスが続き，さらにβシートが来るという構造パターンを表す．

- **judge_keyword_in_authors**: 与えられた文字列を著者名に含むかどうかの真偽値を返す.

2. *Enzyme* クラスは，*Protein* クラスのメソッドを継承する他，次の2つのタイプAのメソッドを持つ.

   - **display_EC_number**: 酵素番号を表示する.
   - **display_all**: 酵素番号を含むすべての文字列型の属性値を表示する．このメソッドは，*Protein* クラスで定義された **display_all** を override する.

3. *EC*.1～*EC*.6，および *Other_enzyme* クラスは，*Enzyme* クラスのメソッドを継承する.

4. *Non_enzyme* クラスは，*Protein* クラスのメソッドを継承する．このクラスに固有のメソッドは無い.

上述のタンパク質を表現するスキーマは，実際のタンパク質の構造を素直に反映しているので，直観的に理解しやすい．しかしながら，データベースの実用性から考察するといくつか問題がある．これについては，3.4節で述べる.

3.3 オブジェクト SQL とメソッド

関係データベースの場合，SQL を用いて検索文を書くが，複雑な計算を必要とするような条件を記述するためには，where 節が長くなる．VERSANT には，オブジェクト SQL (OSQL) が提供されており，関係データベースの SQL と似た構文，即ち select, from, where で検索文を記述する．OSQL では，文の中にメソッドを使用できるので，複雑な計算の部分がメソッドとして提供されていれば，利用者は検索文を簡潔に書くことができる．これを以下の二つの例で説明する.

[例 4] 解像度が 2.0 Å 以下で解析されたタンパク質を検索するという質問を考える．関係データベースで構築されている BIPED に対しては，この検索質問は，SQL を用いて次のように書ける.

```
select   p.PNAME
from     Structure p
where    p.RESOLUTION < 2.0
```

一方，本データベースに対しては，OSQL を用いて次のように書ける．

```
select   p.get_name()
from     Protein p
where    p.get_resolution() < 2.0
```

従って，このようなプリミティブな属性に対する単純な検索質問の場合には，両者の差はない． □

[例 5] "Ala–Gly–Ala" というアミノ酸系列を持つタンパク質を検索する質問を考える．BIPED では，この検索質問は以下のようになる．

```
select S.BRCODE, S.PNAME
from  Structure S, Chain C, Residue R1, Residue R2, Residue R3
where S.BRCODE = C.BRCODE
  and C.BRCODE = R1.BRCODE and C.CHAINID = R1.CHAINID
  and R1.RNAME = 'Ala'
  and C.BRCODE = R2.BRCODE and C.CHAINID = R2.CHAINID
  and R2.RNAME = 'Gly'
  and C.BRCODE = R3.BRCODE and C.CHAINID = R3.CHAINID
  and R3.RNAME = 'Ala'
  and R1.POSITION + 1 = R2.POSITION
  and R2.POSITION + 1 = R3.POSITION
```

このように複雑になるのは，基底関係に保持されている個々のアミノ酸を系列として見るためには，アミノ酸の順序を示す POSITION 属性を用いて処理する必要が生じるためである．VERSANT でも，OSQL だけを用いて検索すると，BIPED の場合と同様に検索文の where 節が非常に複雑なものとなる．しかし，前述の judge_sequence を用いれば，この検索は以下のように簡単に記述できる．

```
select p.get_name()
from   Protein p
where p.judge_sequence("Ala-Gly-Ala")                    □
```

このように，SQL では条件の記述が複雑となる検索の場合でも，OSQL ではメソッドを用いることで，利用者は検索文を簡潔に記述できる．しかし，メソッドは，データベース作成者が C++ などのプログラム言語を用いてそのアルゴリズムをコードしておく必要がある．従って，あらかじめ使用頻度の高い検索を調査し，それらを反映したメソッドを充実させることが重要である．

これらのスキーマを定義しデータベースを構築すれば，VERSANT-Screen というプログラムを用いて，OSQL による検索が対話的に行える．図 3.6 は，上記の検索の実行状況を示している．後ろのウィンドウにはユーザが入力した検索文が，手前のウィンドウには検索の結果が表示されている．

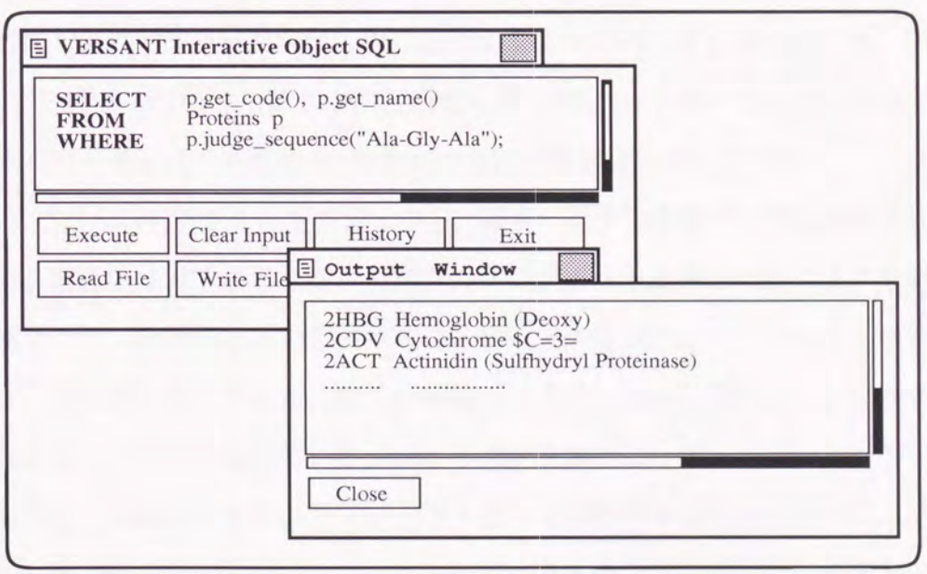


図 3.6 VERSANT-Screen を用いた検索

しかし，OSQL を使うためには，クラス構成，クラスの属性名とメソッド名などを覚えておく必要があり，一般ユーザにとってはまだ使いにくい．そこで，検索されることの多い属性を選び，さらに一般ユーザ向けの検索機能を加えて，グラフィカル・ユーザ・インターフェイス (GUI) を作成した．これについては，3.4.3節で詳細を述べる．

3.4 データベースの実現

3.4.1 原子座標データとデータベースサイズ

オブジェクト指向データベース管理システム VERSANT を用いて，SUN SPARC station 上にデータベースを構築した．メソッドは，C++ を用いてコーディングしている．また，もとの PDB のデータは 1992 年 10 月版で約 280MB であるのに対して，データベースのサイズは，約 500MB となった．この増加の要因は，各原子をそれぞれ 1 つのオブジェクトにしたことによるオーバヘッドと考えられる．なぜなら，1 つの原子のデータ（即ち 3 次元座標）は，80 バイトのレコードであるが，各蛋白質のデータレコード数の 90 %を占めるほど数が多いためである．

さらに，*Atom* クラスのオブジェクト，即ち，原子の座標データの総数は一つのタンパク質あたり数千個となる．これらのオブジェクトを同時に扱うような処理を行う場合には，計算機の能力の大半をキャッシング処理などオブジェクトの管理に費やし，処理効率が悪くなる．原子の座標データは，検索の項目としては通常不要であるため，スキーマを変更して *Atom* クラスを廃止し，原子のデータレコードを全て集めて，圧縮して 1 つのオブジェクトとして蓄積することにした（図 3.7）．圧縮には，汎用性のある UNIX の compress プログラムを用いた．この結果，データベースの容量は 200MB まで小さくできた．さらに，蛋白質の立体構造を描くプログラムには，原子のデータを圧縮された形で渡すことができるため，データの受け渡しが簡単で，かつ高速になる．

3.4.2 アミノ酸系列に対する検索の効率化

アミノ酸系列に対する検索は頻繁に実行され，その検索にはアミノ酸の名前だけが必要である．この検索処理には，各タンパク質の鎖ごとに *Residue* クラスの全オブジェクトを参照し，アミノ酸名を連接することでアミノ酸系列を生成しなければならない．この際に数百個のオブジェクトを参照することになり，そのオーバヘッドのために検索速度が遅くなる．これは，全ての残基オブジェクトを，順次，ディスクからメモリに一旦読込むが，データベース管理システムとして用意したキャッシュ領域（10MB に設定時）に入り切らないためである．

この解決策として，*Residue* クラスとは別に，*Residue − sequence* という新しいクラスを設け，そこに，鎖を構成しているアミノ酸の名前だけを文字列として蓄積するようにした（図 3.7）．情報は冗長になるが，検索時間を大幅に減らすことができた．この文字列を持たせることは，関係データベースでの結合操作で得られた関係を蓄積することに相当する．

オブジェクトが多数のオブジェクトで構成されるときには，スキーマ設計時に，このような工夫が必要となる．

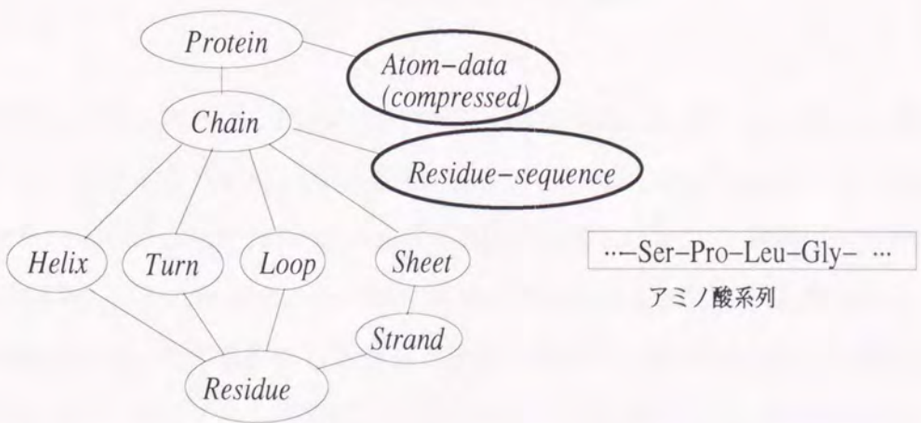


図 3.7 スキーマ修正後の複合オブジェクトの構成

3.4.3 グラフィカル・ユーザ・インターフェイス

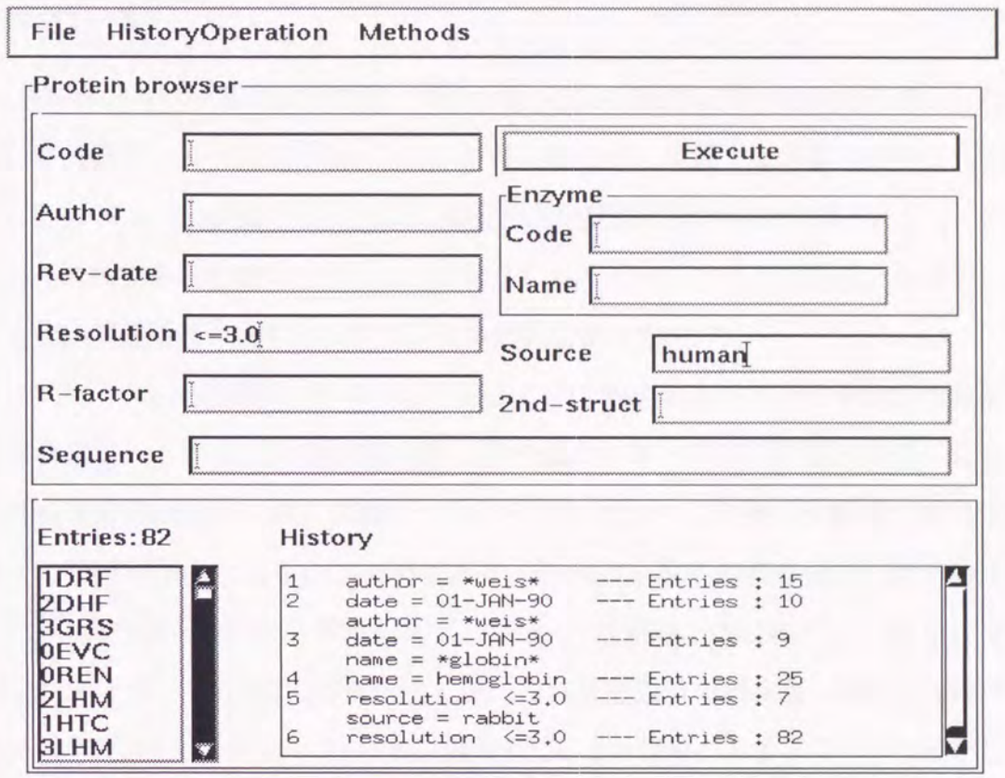


図 3.8 *Protein Browser* 画面

グラフィカル・ユーザ・インターフェイス (GUI) として，*Protein Browser*を 提供している．GUI の作成には，米国 NIH （National Institute of Health）の National Center for Biotechnology Information (NCBI) で開発された GUI 構築ツール Vibrant を用いている．*Protein Browser*では，タンパク質のいくつかの項目について，条件の入力を求める．その条件にしたがって検索を実行し，得られたタンパク質のエントリ名をリストアップし，さらに，利用者によって選ばれたエントリについて，詳細な属性を表示する．また，過去の検索結果を保存したり，検索結果に対して和集合や積集合を取ったり，PDB データを利用する応用プログラムを起動したりできる．図 3.8は， 画面の例である．*Protein Browser* で提供する主な機能を次節で示す．

3.4.4 仮想クラス機能と履歴機能

仮想クラス機能

関係データベースにおいては，ビューがSQLの規格に盛り込まれており，たいていの関係データベース管理システムでは，ビューを定義する機能が提供されている．ビューは，仮想的な関係であり，複数の基底関係に対する検索質問として，データベース管理者または利用者により定義される．ビューをマクロ記述として用いることで，基底関係に対する検索文が簡単に記述できる．

オブジェクト指向データベースにおいては，特定のクラスだけを検索対象にする場合には，そのクラスを検索対象に選べばよいが，クラス階層と適合しないような検索条件の場合には，関係データベースのビューに相当する機構があれば検索文の記述が簡単になるので便利である．しかし，著者の知る限り，商用のオブジェクト指向データベース管理システムでビューを提供しているのは，O_2 だけであり [Mamou 92, Abit 91]，研究者間で統一された定義がまだ存在しない [Rund 92]．神戸大学の田中は，Smalltalkの言語環境でビューを提供する方法について提案している [Tana 89]．そこでは，「ビュー・クラス」という新しいクラスを一つ設け，各ビューの定義をインスタンスとして保持する．

本データベースでも，ビュークラスを準備し，インスタンスとして一種のビュー（ここでは仮想クラスと呼ぶ）を保持できるようにする．但し，本システムでは，仮想クラスは仮想的な関係を定義するというより，むしろ検索処理の効率を良くすることを目的として考える．例えば，「解像度が2.0以下の酵素タンパクを求める」という検索では *Enzyme* クラスおよびそのサブクラスに対して検索し，*Non_enzyme* クラスは検索対象にしなくて良い．即ち，クラス階層が頻繁に実行される検索に沿うものであれば，効率良く検索することができる．しかし，スキーマ設計者の持つ観点とデータベース利用者の持つ観点は異なることが多い．例えば，「採取源がヒトであり，かつ解像度が2.0以下のタンパク質を求める」という検索では，*Protein* クラ

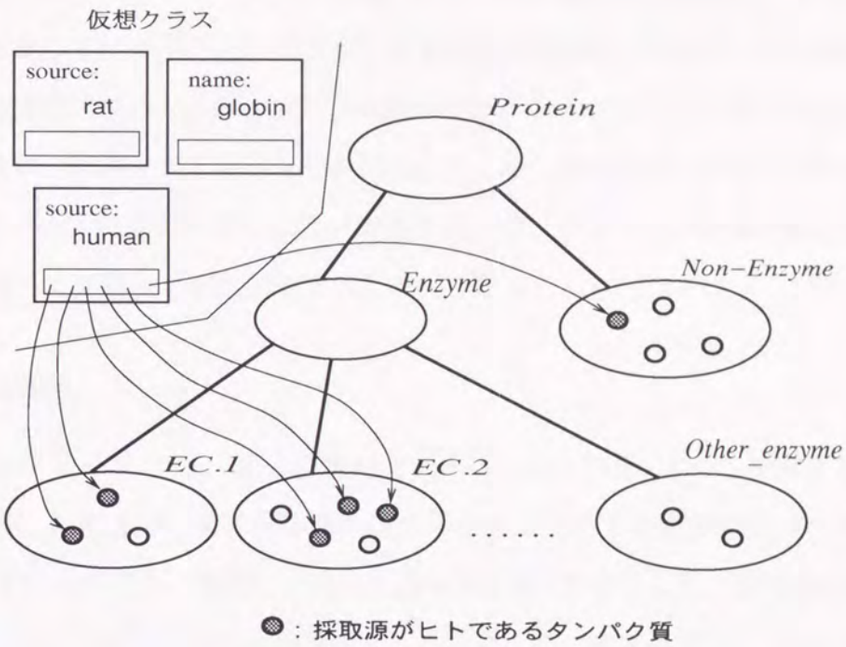


図 3.9 仮想クラスの定義

スおよびそのすべてのサブクラスを検索する必要がある．そこで，実行頻度の高い検索に対して仮想クラスをあらかじめ設定し，各仮想クラスで定義された条件を満たすオブジェクトのオブジェクト ID の集合も保存する．図 3.9では，採取源によって，いくつかの仮想クラスを定義し，オブジェクト ID を保存している様子を示している.「採取源がヒトであり，かつ解像度が 2.0 以下のタンパク質を求める」という検索をしたい場合には，利用者は，この「採取源がヒトであるタンパク質」の仮想クラスに対して,「解像度が 2.0 以下」という条件で検索質問を適用すれば，探索の対象領域が狭められるので，検索が効率良く実行される．

また，利用者は，自分自身が設定した仮想クラスを用いて検索を行うことにより，検索を効率化良く実行することができる．*Protein Browser*では，この仮想クラスの機能を用いて，検索結果を保存する機構を実現している．

仮想クラスは，*is-a* 関係から言えば，クラス階層の根のクラスのサブクラスと

いえるが，仮想クラスのインスタンスにメソッドを適用する場合は，インスタンスが実際に属しているクラスのメソッドを選ぶ必要がある．例えば，*Enzyme* クラスにおいて定義される **display_EC_number** というメソッドは，*Non_enzyme* には存在しない．仮想クラスのインスタンスの中には，*Non_enzyme* クラスに属すオブジェクトの ID が含まれる可能性があるため，オブジェクトが実際に属しているクラスを調べて，適用可能か判定するようにした．

検索履歴機能

Versant Screen では，過去に実行された Object SQL による質問文を保持し，それを修正して次の検索文を簡単に記述できるようにする機能が提供されている．*Protein Browser* でも，複数回の検索を効率的に実行できるように検索履歴機能を実現した．前述の仮想クラス機能を用いることにより，検索履歴として検索条件だけではなく，検索結果のオブジェクト ID も含めて保持する．この機能により過去に実行された検索結果を再表示したり，新たに条件を付加した検索を効率良く実行できる．

例えば，「1992 年以降に登録されて，かつ解像度が 2.0 以下のタンパク質」を求めるという検索は，先に「1992 年以降に登録されたタンパク質」を求める検索が実行されていれば，利用者は履歴を利用して，検索結果の中からさらに「解像度が 2.0 以下のタンパク質」を検索すればよい．あるいは，「解像度が 2.0 以下のタンパク質」も検索されていれば，「1992 年以降に登録されたタンパク質」との集合積を求めて目的とする検索結果をすぐに得ることができる．

3.4.5 応用プログラムとのリンク

これまでに PDB に対して，立体構造の表示や構造分析を行う応用プログラムがいくつか開発されてきた．検索で得られたタンパク質に対して，これらの既存の応用プログラムを起動させたいことがよくあるので，応用プログラムを起動するメ

ソッドを用意している．3.4.2節で述べたように，PDB のファイル毎にデータを圧縮して格納しているので，メソッド内でデータを展開 (uncompress) して PDB フォーマットに戻し，既存のプログラムに渡せば良い．

三次元立体表示については，RasMol という応用プログラムがあり，ボール・モデル，リボン・モデル，ワイヤフレーム・モデルなど各種の形式で立体構造が表示できる．図 3.10は，ヒトのヘモグロビンの立体構造をボール・モデルで表示したものである．このように図示することによって，利用者は立体構造を直観的に把握できる．なお，RasMol は，圧縮データを自分で伸長する機能があるため，本データベースから圧縮データをそのまま渡すことができ効率が良い．

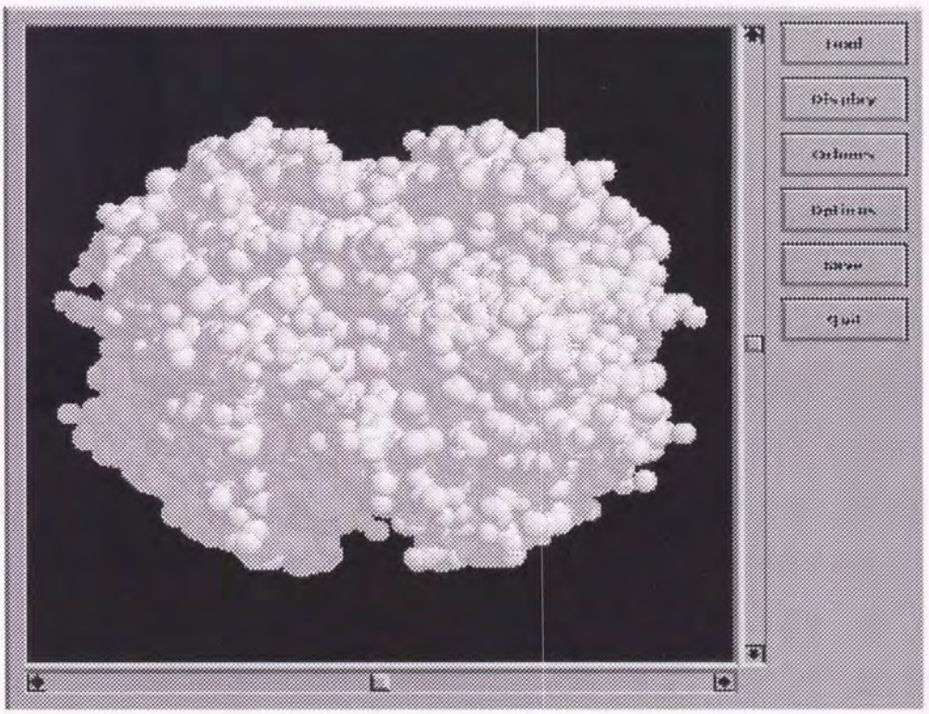


図 3.10 RasMol によるヘモグロビンの立体表示

RasMol 以外では，原子座標のデータを用いて 2 次構造を計算によって求める DSSP [Kab 83] という応用プログラムが良く用いられるので，本システムから起動

できるようにしている.

3.4.6 更新について

ユーザがデータベースを更新することは無いので，更新操作は提供しない．従って，仮想クラスを通した更新という難しい問題も考える必要はない．さらに，PDBの更新は，1年に4回全体が一括して更新され，短い周期で更新の差分が配布されることはない．したがって，データベースは，PDBが新しくなるたびに，全体を作成しなおせばよいと考えている．また，各仮想クラスには，オブジェクト識別子だけでなく，その仮想クラスを生成するための情報も保持しているので，PDBの更新情報が差分データとして提供されるようになっても対応が可能である．

3.5 関連研究

3.5.1 インデックスと仮想クラスの比較

筆者の知る限りでは，オブジェクト指向データベースにおいて，検索処理効率を上げるためにデータに冗長性を持たせるスキーマ設計手法は研究されていなかった．本データベースの仮想クラスは，一見単純なインデックスに似ている．しかし,「採取源」のような単純な文字列属性に対してだけでなく，超二次構造として“$\beta-\alpha-\beta$”構造を持つかどうかを調べる**judge_structure**などのメソッド群を予め実行し，該当するタンパク質のOIDをそれぞれ仮想クラスとして持たせることにより，利用者の視点によるクラス分類を疑似的に提供することができる．また，予め検索を実行してそのインデックスを作るという方法は，関係データベースで提案された*creative index* [Inmon 92]と共通した考え方と言える．

オブジェクト指向でのインデックスの研究としては，よく用いられる検索のパスに対して，インデックスを作成するものがある[Maier 86, Kemp 90]．パス表現とは，

emp を従業員のオブジェクト，*dept* を所属部署の OID を返す属性，*manager*を管理者の OID を返す属性，*name* を名前を返す属性とするとき，*emp.dept.manager.name* のような形式の表現であり，オブジェクトをたどって従業員の管理者名を検索することを表す．このパスに現れる複数の属性の値（OID）をインデックスに持たせて，検索の効率を向上させようという試みである．また，関係データベースにおける *materialized view* [Mamou 92, Bert 92] では，ビューという仮想的な関係を永続的な関係としてディスクに保存するのに対して，本論文での仮想クラスは，オブジェクトの複製を作成・保存するのではなく，仮想クラスの定義と OID だけを保存しているところが異なる．

3.5.2 オブジェクト指向のタンパク質データベース

オブジェクト指向データベースを用いたタンパク質データベースとして，最初に試作されたものは P/FDM である [Gray 90a, Gray 90b, Gray 92]．本システムと同様，タンパク質の構造を素直にスキーマに反映させている．P/FDM との違いは以下の 3 点である．

- P/FDM では，タンパク質はクラス分けされずに一つのクラスに入れられている．本システムでは，図 3.3に示すように，8 つのクラスに分類しているので，酵素のクラスを指定した検索は，処理効率が良くなる．
- 本システムでは，仮想クラスを提供している． 3.4節で述べたように，仮想クラスを使うことによって，検索の対象となるオブジェクトの数が制限されるので，検索処理が高速になる．
- P/FDM では，Prolog が用いられているので，推論機能を生かした検索が行える．

この他に，試作中のデータベースとして OOPDB (Object-Oriented Toolbox for the Protein Data Bank) があり．PDBlib と PDBtool が開発されている [Chang 94,

Pu 92]. PDBlib は，オブジェクト指向言語 C++ のクラスライブラリであり，PDB ファイルから読み込んだタンパク質の構造を，メモリ上にオブジェクト指向で表現する．PDB のデータフォーマットが，後述する CIF フォーマットに変更されても，PDBlib のモジュールを一部変更することによって対応できるという特徴がある．PDBtool は，新しいタンパク質の設計を支援するためのツールであり，グラフィカル・ユーザ・インターフェイスを通して，立体構造を変更し検証する機能を提供している．現在，PDBlib でメモリ上に作成されたオブジェクトを，オブジェクト指向データベース管理システム ObjectStore を用いて，データベースに蓄積する方向で研究が進められている．

3.6 むすび

商用のオブジェクト指向データベース管理システム VERSANT を用いて試作したタンパク質のデータベースについて紹介し，スキーマ設計について工夫した点を中心に述べた．

まず，アミノ酸の系列データに対する検索については，オブジェクトをたどるとディスク I/O が増えてしまうため，検索に必要なアミノ酸の名前だけを文字列データとしてデータ登録時に作成しておくことにより，冗長性は増すものの，検索が高速に行えることを確認した．この方法は，タンパク質データの更新が殆んどないという前提があるので採用できた．一般に，科学技術データベースでは，データの追加が殆んどで更新が少ないという性質があるものの，順序をもったデータ，例えば，系列，リスト，多次元配列，グラフなどを扱う必要があり，関係データベースでもオブジェクト指向データベースでもこれらを扱う適切なデータ構造が用意されていない [Maier 93]．この問題を根本的に解決するためには，さらに研究が必要である．

また，検索処理を高速化するための別の方法として，仮想クラスを考案した．検索の多い属性についてオブジェクトの OID を保存したり，メソッドを実行して，条

件に合致するオブジェクトのOIDを保存することが，仮想クラスという形で統一して扱える．さらに，仮想クラスを利用して，検索履歴を保存することも可能であるため，検索の処理効率を向上させることが出来る．一方，仮想クラスを用いて検索してほしい場合，利用者がOSQLの検索文に，仮想クラスを陽に指定する必要がある．仮想クラスの存在を利用者に意識させることなく，検索条件から最適な仮想クラスを自動的に判断して用いる機構も必要である．データを追加する場合は，仮想クラスの定義をしている検索文を実行すればよいが，更新を考える場合には，自動的に仮想クラスを無効に(invalidate)するような機構が必要である．これらは，今後の研究課題である．

VERSANTをはじめとしてONTOS，Objectivity/DBなど，現在の商用のオブジェクト指向データベース管理システムはC++を基礎にしているものが多いので，それらを用いた場合にも，本論文で述べたスキーマ構成や仮想クラスの構築方法を適用することが可能である．最後に，本データベースをタンパク質構造データベースとして実用化する場合の課題を以下に挙げる．

- PDBファイルからの変換

  PDBのデータには，アミノ酸系列番号が抜けていたり間違って記述されている場合が散見される．また，検索に必要なデータを，自然語による文章で書かれた項目から抽出して，本データベースのデータ項目にする作業はなかなか困難である．本データベースは試作であり，全データを洩れなく登録しているわけではないので，実用化のためには，上記の問題点を手作業的に行ってデータを入力する必要がある．

  現在，PDBの表記法を改善するための活動が行われており，Crystallographic Information File (CIF)やAbstract Syntax Notation One (ASN.1)を用いた記述フォーマットの規格作りが進められている．標準化後には，上述のような変換の問題が解消されるものと期待できる．

- 同義語辞書

  タンパク質名や採取源名などでは，同一のものに様々な呼称が存在する場合がある．同義語を判断する辞書を利用することにより，検索機能を向上させることが期待できる．

上記の課題を解決するほかに，よく使用する検索をメソッドとして準備したり，リンクする応用プログラムの種類を充実させることも，タンパク質の研究者に使用してもらう場合には必要となる．

第 4 章

結辞

本論文では，検索の処理効率を向上させることを狙いとしたデータベーススキーマの設計・構築手法について，理論と実際の両面から研究した結果をまとめた．

まず，第 2 章で関係データベースの論理設計に関して述べた．関係データベースは，理論的にしっかりした裏付けがあり，SQL の規格に準拠した商用のデータベース管理システムが数多く利用されるようになってきた．しかし，正規化によって基底関係が多くなるために，期待された検索の性能が出ないという問題も実際のフィールドにおいて発生している．スキーマ設計を工夫することによって，この問題を解決しようという提案は行われてきたが，理論的な議論は行われていなかった．本論文では，スキーマ変形を行っても，不整合を生じないための必要十分条件を明らかにした．さらに，その必要十分条件が成立しない場合でも，SQL のトリガー機能を用いるなどすれば，条件を緩められることを示した．本論文の結果から，スキーマ変形を行っても良いかどうかは機械的に判定できるようになり，不整合の生じないデータベースを実現することが可能になる．

第 3 章では，実際にオブジェクト指向データベースを作成して得たスキーマ設計に関する知見を中心に述べた．科学技術データは更新されることが少ないという点に着目し，データの例としてタンパク質構造データを選び，商用のオブジェクト

指向データベース管理システムを用いてタンパク質データベースを試作した．タンパク質を酵素の種類によって分類しクラス階層を構成し，各タンパク質を複合オブジェクトで表現することにより，タンパク質の構造を素直にモデル化できる．しかし，検索処理を高速にする為には，冗長なデータを持たせることや，あらかじめ検索質問を実行した得られたオブジェクトの OID を仮想クラスとして保存するなど，スキーマ設計上の工夫が必要であった．酵素によるクラス分類は管理者の視点であるが，仮想クラスを利用することによって，管理者とは異なった利用者の視点に従って，クラスを疑似的に分類することも可能となる．しかし，検索の処理効率を向上させることを目的としているので，仮想クラス間の階層構造は考慮していない．本論文では，商用のオブジェクト指向データベース管理システムを用いてデータベースを実現したが，C++言語に基づいたデータベース管理システムを用いる場合には，同じ設計手法を適用することが可能である．

今後の研究方向としては，現実の生物情報関連のデータベースを扱う上の問題を考えたい．具体的には，タンパク質立体構造の 3 次元マッチングを効率的に処理できるようなデータベーススキーマ構造や，遺伝子データベースに対する近似文字列検索を高速に実行するためのスキーマ構造について検討していく予定である．

謝辞

本研究の全過程を通じて，御指導頂いた大阪大学基礎工学部情報工学科橋本昭洋教授に感謝します．

関係データベースの論理設計に関して，詳細に渡り親切に御指導頂いた滋賀大学経済学部情報管理学科葛山善基助教授に感謝します．

本研究のみならず，著者が各種の論文をまとめるにあたり常に詳細に渡り御指導頂いている奈良先端科学技術大学院大学伊藤実教授に感謝します．

本論文をまとめるにあたって御教示を頂いた大阪大学基礎工学部情報工学科都倉信樹教授，萩原兼一教授，松田秀雄助教授，工学部情報システム工学科西尾章治郎教授に感謝します．

本研究の遂行にあたり，常に暖かく見守り御鞭撻頂いた奈良先端科学技術大学院大学嵩忠雄教授に感謝します．

タンパク質データベースの設計に関して，助言を頂いた大阪大学蛋白質研究所楠木正巳助手に感謝します．また，タンパク質データベースの設計・構築に協力して頂いた橋本研究室の片野宏一氏（現在，富士通勤務），地村篤氏（現在，菊野研究室），吉田正宏氏，米田真治氏に感謝します．

最後になりましたが，橋本研究室の皆様に感謝します．

参考文献

[Abit 91] Abiteboul, S. and Bonner, A.: "Objects and views," SIGMOD, pp.238-247 (1991).

[Bern 77] Bernstein, F.C., Koetzle, T.F., Williams, G.J.B., Meyer, D.F., Brice, M.D., Rodgers, J.R., Kennard, O., Shimanouchi, T., and Tasumi, M. : "The Protein Data Bank: A computer-based archival file for macromolecular structures," Journal of Molecular Biology, vol. 112, pp.535-542 (1977).

[Bert 92] Bertino, E.: "A view mechanism for object-oriented databases," 3rd International Conference on Extending Database Technology, Lecture Notes in Computer Science #580, pp.136-151 (1992).

[Chang 94] Chang, W., Shindyalov, I.N., Pu, C., and Bourne, P.E.: "Design and application of PDBlib, a C^{++} macromolecular class library," To appear in Computer Applications in the Biosciences (1994).

[Clark 90] Clark, D.A., Barton, G.J., and Rawlings, C.J.: "A knowledge-based architecture for protein sequence analysis and structure prediction," Journal of Molecular Graphics, vol. 8, pp.94-107 (1990).

[Erick 92] Erickson, D.: "Hacking the genome," Scientific American, pp.98-105 (Apr. 1992), 中西真人訳："データベース化に悩むヒトゲノム計画," 日経サイエンス (1992.6).

[Gray 90a] Gray, P.M.D., Paton, N.W., Kemp, G.J.L., and Fothergill, J.E.: "An object-oriented database for protein structure analysis," Protein Engineering, vol. 3, no. 4, pp.235-243 (1990).

[Gray 90b] Gray, P.M.D. and Kemp, G.J.L.: "Finding hydrophobic microdomains using an object-oriented database," Computer Applications in the Biosciences, vol. 6, no. 4, pp.357-363 (1990).

[Gray 92] Gray, P.M.D., Kulkarni, G.K., and Paton, N.W.: "Object-oriented databases: A semantic data model approach," Prentice Hall (1992).

[Huys 91] Huysmans, M., Richelle, J., and Wodak, S.J.: "SESAM: A relational database for structure and sequence of macromolecules," Proteins, vol. 11, pp.59-76 (1991).

[Inmon 87] Inmon, W. : "Denormalization of data," Proceedings of the 12th Structured Method Conference, pp.148-165 (1987).

[Inmon 89] Inmon, W. : "DB2: Maximizing performance of online production systems," QED Technical Publishing, pp.207-222 (1989).

[Inmon 92] Inmon, W. : "Building the data warehouse," John Wiley & Sons, pp.91-99 (1992).

[Inoue 93] 井上望："データベース設計," 日経オープンシステム, no. 5, pp.93-110, 日経 BP 社 (1993).

[Islam 89] Islam, S.A. and Sternberg, M.J.E.: "A relational database of protein structures designed for flexible enquiries about conformation," Protein Engineering, vol. 2, no. 6, pp.431-442 (1989).

[Kab 83] Kabsch, W. and Sander, C.: "Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features ," Biopolymers, vol. 22, pp. 2577-2637 (1983).

[Kata 93] 片野宏一，中西通雄，伊藤実，橋本昭洋："タンパク質3次構造データのオブジェクト指向データベースの試作," 情報学シンポジウム, pp.19-28 （1993）.

[Kemp 90] Kemper, A. and Moerkotte, G.: "Advanced query processing in object bases using access support relations," Proceedings of 16th International Conference on Very Large Data Bases, pp.290-301 (1990).

[Kirk 92] Kirkwood, J. : "High performance relational database design," Ellis Horwood, pp.142-146 (1992).

[Kitsu 91] 喜連川優, 中野美由紀："機能ディスクシステム：関係データベース処理とその性能評価," 信学論 (D-I), J74-D-I, 8, pp.496-507 (1991).

[Korth 91] Korth, H. and Silberchatz, A. : "Database system concepts (Second Edition)," McGraw-Hill (1991).

[Kuha 91] Kuhara, S., Satou, K., Furuichi, E., Takagi, T, Takehara, H. and Sakaki, Y.: "A deductive database system PACADE for the three dimensional structure of protein," Proceedings of 24th Hawaii International Conference on System Sciences, pp.653-659 (1991).

[Maier 83] Maier, D. : "The theory of relational databases," Computer Science Press (1983).

[Maier 86] Maier, D. and Stein, J. : "Indexing in an object-oriented DBMS," Proceedings of IEEE International Conference on Object-Oriented Database Systems, pp.171-182 (1986).

[Maier 93] Maier, D. and Vance, D. : "A Call to order," Proceedings of ACM PODS, pp.1-16 (1993).

[Maki 77] Makinouchi, A. : "A consideration on normal form of not-necessarily-normalized relation in the relational data model," Proceedings of the third International Conference on Very Large Data Bases, pp.447-453 (1977).

[Mamou 92] Mamou, J.-C. and Medeiros, C.B.: "Interactive manipulation of object-oriented views," Proceedings of 7th International Conference on Data Engineering, pp.60-69 (1991).

[Masu 94] 増永良文："ダウンサイジングとデータベース" など, 特集「ダウンサイジング時代のデータベース」, Computer Today, no. 60, pp.6-41, サイエンス社 (1994).

[Naka 90] 中西通雄, 葛山善基, 伊藤実, 橋本昭洋："事務処理に向いた関係データベースのスキーマ設計手法についての一考察," 信学技法, DE90-23, pp.33-40 (1990)

[NTT 93] 日本電信電話（株）情報通信研究所:"データベース論理設計," pp.27-47, 阿部出版 (1993).

[Patt 90] Pattabiraman, N., Namboodiri, K., Lowrey, A., and Gaber, B.P.: "NRL_3D: a sequence-structure database derived from the protein data bank(PDB) and searchable within the PIR environment," Protein Sequences and Data Analysis, vol. 3, pp.387-405 (1990).

[Pu 92] Pu, C., Sheka, K.P., Ong, J., Chang, L., Chang, A., Alessio, E., Shindyalov, I.N., Chang, W., and Bourne, P.E.: "PDBtool: A prototype object oriented toolkit for protein structure verification," Technical Report CUCS-048-92, Department of Computer Science, Columbia University (1992).

[Rund 92] Rundensteiner, E.A.: "Multi-View: A methodology for supporting multiple views in object-oriented databases," Proceedings of 18th International Conference on Very Large DataBases, pp.187-198 (1992).

[Satou 94] Satou, K., Furuichi, E., Takiguchi, K., Kuhara, S. and Takagi, T. : "Application of a deductive database system PACADE toward discovery of clusters of similar structures in proteins," Proceedings of 27th Hawaii International Conference on System Sciences, pp.160-169 (1994).

[Sayle 92] Sayle, R. and Bissell, A.: "RasMol: A program for fast realistic rendering of molecular structures with shadows," Proceedings of 10th Eurographics Conference (1992).

[Sch 81] Schkolnick, M.: "The effects on denormalization on database performance," IBM Reseach Report RJ3082 (38128), pp.1-17 (1981)

[Shos 92] Shoshani, A.: "A layered approach to scientific data management at Lawrence Berkeley Laboratory," Bulletin of the Technical Committee on Data Engineering, vol. 16, no. 1, pp.4-8 (1993).

[Taka 92] 高木利久："ゲノムデータベース," 情報処理, vol.33, no.10 (1992)

[Tana 89] Tanaka, K., Yoshikawa, M., and Ishihara, K.: "Schema design, views and incomplete information in object-oriented databases," Journal of Information Processing, vol. 12, no. 3, pp.239-250 (1989).

[Tana 93] 田中克己："サイエンティフィックデータベースの新たな展開," 理研シンポジウム「データベースにおける変革」講演要旨集, pp.46-54 (1993) .

[TCDE 93] IEEE Computer Society: "Special issue on scientific databases," Bulletin of the Technical Committee on Data Engineering, vol. 16, no. 1 (1993).

[Tsuji 91] 辻貞孝：“関係データベーススキーマの再編による検索処理効率改善効果の評価,” 大阪大学基礎工学部情報工学科特別研究報告（卒業論文）(1991)．

[Yoshi 94] 吉田正宏，米田真治，中西通雄，橋本昭洋：“タンパク質データベースの試作 - 検索機能について”，信学技報，DE94-8, pp.53-60（1994）．

付録

A. 属性一覧

Protein, *Chain*, *Helix*などのクラスの属性を示す.

*Protein*クラス

- **Name**：タンパク質の名称
- **Functional_Classification**：タンパク質のコンパウンド名
- **Authors**：著者オブジェクトへのポインタ
- **Structure_Analysis_Method**：タンパク質の立体構造の解析に用いた手法名
- **Resolution**：立体構造の解析に用いたX線の波長
- **Refinement**：精製の手法
- **Source**：タンパク質の採取源（種，臓器，組織，変種）
- **Journal**：このタンパク質の構造について発表した文献
- **References**：上記の文献以外で，この構造に関連する重要な文献
- **Remarks**：コメント欄（PDBのREMARKレコード）
- **Sites**：タンパク質の活性部位，特殊な機能を持つアミノ酸の部位
- **Code**：タンパク質のPDBでのファイル名
- **Heteros**：タンパク質に含まれる非アミノ酸（金属，ヘムなど）
- **Number_of_Chains**：タンパク質を構成する鎖の数
- **Chains**：鎖オブジェクトへのポインタ
- **Date**：PDBにおけるデータの登録日
- **Revision_Date**：PDBにおけるデータの改訂日
- **Footnotes**：特定の原子やアミノ酸に関する注釈
- **Scale**：Å単位の標準座標からfractional座標値への変換値
- **Cryst**：cell定数の値
- **Origx**：Å単位の標準座標から投稿された座標値への変換値
- **Mtrix**：非結晶シンメトリを表す変換値

- **Tvect**：共有結合の変換ベクトル
- **Connects**：一次構造以外の結合情報

要素オブジェクトのクラス属性を以下に示す.

1. *Chain*クラス
   - **Chain_id**：鎖のID（A, B 等）
   - **Number_of_Residues**：鎖を構成するアミノ酸の数
   - **Number_of_Helixes**：鎖の中のヘリックスの数
   - **Number_of_Turns**：鎖の中のターンの数
   - **Number_of_Loops**：鎖の中のループの数
   - **Number_of_Sheets**：鎖の中の β シートの数
   - **Helixes**：ヘリックスオブジェクトへのポインタ
   - **Turns**：ターンオブジェクトへのポインタ
   - **Loops**：ループオブジェクトへのポインタ
   - **Sheets**：シートオブジェクトへのポインタ
   - **Sequences**：先頭のアミノ酸オブジェクトへのポインタ
   - **Residue-Sequence**：アミノ酸系列（文字列）オブジェクトへのポインタ
2. *Helix*クラス
   - **Number_of_Residues**：ヘリックスを構成するアミノ酸の数
   - **Type_of_Helix**：ヘリックスの型（ α-helix, π-helix 等）
   - **First_Residue**：ヘリックスを構成する先頭のアミノ酸オブジェクトへのポインタ
   - **Last_Residue**：ヘリックスを構成する最後のアミノ酸オブジェクトへのポインタ
3. *Turn*クラス
   - **Number_of_Residues**：ヘリックスを構成するアミノ酸の数
   - **First_Residue**：ターンを構成する先頭のアミノ酸オブジェクトへのポインタ
   - **Last_Residue**：ターンを構成する最後のアミノ酸オブジェクトへのポインタ
4. *Loop*クラス
   - **Number_of_Residues**：ループを構成するアミノ酸の数
   - **First_Residue**：ループを構成する先頭のアミノ酸オブジェクトへのポインタ

- **Last_Residue**：ループを構成する最後のアミノ酸オブジェクトへのポインタ

5. *Sheet*クラス
   - **Number_of_Strands**：シートを構成するストランドの数
   - **Strand**：ストランドオブジェクトへのポインタ
6. *Strand*クラス
   - **Number_of_Residues**：ストランドを構成するアミノ酸の数
   - **Parallel**：ストランド方向（直前のストランドと順方向か逆方向か）
   - **First_Residue**：ストランドを構成する先頭のアミノ酸オブジェクトへのポインタ
   - **Last_Residue**：ストランドを構成する最後のアミノ酸オブジェクトへのポインタ
7. *Residue*クラス
   - **Name**：アミノ酸の名前
   - **Position**：鎖の中での位置（順序番号）
   - **Ssbond**：硫黄結合をしているアミノ酸オブジェクトへのポインタ
   - **Number_of_Atoms**：アミノ酸を構成している原子数

B. メソッド一覧

ここでは，3.2.3節で紹介したメソッドの内，タイプCのメソッドを列挙する．

- **judge_keyword_in_name**：タンパク質の名前に，与えられたキーワードが含まれるか否かを判定する．
- **judge_keyword_in_title**：参考文献のタイトルに，与えられたキーワードを含むものがあるか否かを判定する．
- **judge_keyword_in_authors**：著者名に，与えられたキーワードを含む著者があるか否かを判定する．
- **judge_keyword_in_source**: 採取源に，与えられたキーワードを含むものがあるか否かを判定する．
- **judge_keyword_in_refinement**: 精製手法に，与えられたキーワードが含まれるか否かを判定する．
- **judge_residue_in_site**: 活性部位に，与えられたキーワードを含むアミノ酸があるか否かを判定する．
- **judge_hetero_in_protein**: ヘテロ分子に，与えられたキーワードを含む分子があるか否かを判定する．

- **judge_sequence:** アミノ酸系列に，与えられたパターンの部分系列を含むか否かを判定する.
- **judge_sequence_in_helix, judge_sequence_in_loop, judge_sequence_in_turn, judge_sequence_in_strand:** 2次構造を構成するアミノ酸系列に，与えられたパターンの部分系列を含むか否かを判定する.
- **judge_structure:** タンパク質が，与えられた2次構造のパターンを含むか否かを判定する.